

REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-98-

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

thering
tion of
r, Suite

0784

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE December, 1994	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE USAF Summer Research Program - 1994 High School Apprenticeship Program Final Reports, Volume 14, Rome Laboratory		5. FUNDING NUMBERS
6. AUTHORS Gary Moore		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research and Development Labs, Culver City, CA		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NI 4040 Fairfax Dr, Suite 500 Arlington, VA 22203-1613		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES Contract Number: F49620-93-C-0063		
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release		12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words) The United States Air Force High School Apprenticeship Program's (USAF- HSAP) purpose is to place outstanding high school students whose interests are in the areas of mathematics, engineering, and science to work in a laboratory environment. The students selected to participate in the program work in an Air Force Laboratory for a duration of 8 weeks during their summer vacation.		
14. SUBJECT TERMS AIR FORCE HIGH SCHOOL APPRENTICESHIP PROGRAM, APPRENTICESHIP, AIR FORCE RESEARCH, AIR FORCE, ENGINEERING, LABORATORIES, REPORTS, SCHOOL, STUDENT, SUMMER, UNIVERSITIES		15. NUMBER OF PAGES
		16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified
20. LIMITATION OF ABSTRACT UL		

UNITED STATES AIR FORCE
SUMMER RESEARCH PROGRAM -- 1994
HIGH SCHOOL APPRENTICESHIP PROGRAM FINAL REPORTS

VOLUME 14

ROME LABORATORY

RESEARCH & DEVELOPMENT LABORATORIES

5800 Uplander Way

Culver City, CA 90230-6608

Program Director, RDL
Gary Moore

Program Manager, AFOSR
Major David Hart

Program Manager, RDL
Scott Licoscas

Program Administrator, RDL
Gwendolyn Smith

Program Administrator, RDL
Johnetta Thompson

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Bolling Air Force Base

Washington, D.C.

December 1994

19981204 036

PREFACE

Reports in this volume are numbered consecutively beginning with number 1. Each report is paginated with the report number followed by consecutive page numbers, e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3.

This document is one of a set of 16 volumes describing the 1994 AFOSR Summer Research Program. The following volumes comprise the set:

VOLUME

TITLE

1	Program Management Report
	<i>Summer Faculty Research Program (SFRP) Reports</i>
2A & 2B	Armstrong Laboratory
3A & 3B	Phillips Laboratory
4	Rome Laboratory
5A & 5B	Wright Laboratory
6	Arnold Engineering Development Center, Frank J. Seiler Research Laboratory, and Wilford Hall Medical Center
	<i>Graduate Student Research Program (GSRP) Reports</i>
7	Armstrong Laboratory
8	Phillips Laboratory
9	Rome Laboratory
10	Wright Laboratory
11	Arnold Engineering Development Center, Frank J. Seiler Research Laboratory, and Wilford Hall Medical Center
	<i>High School Apprenticeship Program (HSAP) Reports</i>
12A & 12B	Armstrong Laboratory
13	Phillips Laboratory
14	Rome Laboratory
15A&15B	Wright Laboratory
16	Arnold Engineering Development Center

HSAP FINAL REPORT TABLE OF CONTENTS

i-xiv

1. INTRODUCTION	1
2. PARTICIPATION IN THE SUMMER RESEARCH PROGRAM	2
3. RECRUITING AND SELECTION	3
4. SITE VISITS	4
5. HBCU/MI PARTICIPATION	4
6. SRP FUNDING SOURCES	5
7. COMPENSATION FOR PARTICIPANTS	5
8. CONTENTS OF THE 1994 REPORT	6

APPENDICIES:

A. PROGRAM STATISTICAL SUMMARY	A-1
B. SRP EVALUATION RESPONSES	B-1

HSAP FINAL REPORTS

SRP Final Report Table of Contents

Author	University/Institution Report Title	Armstrong Laboratory Directorate	Vol-Page
Eugenia D Baker	A. Crawford Mosley High School , Lynn Haven , FL Reinventory of the Technical Information Center of	AL/EQP	12 - 1
Sara E Berty	Carroll High School , Dayton , OH The Biological Effects of an ADN on Hepatocytes:	AL/OET	12 - 2
Michael J Bruggeman	Archbishop Alter High School , Kettering , OH Cardiac Measures of Pilot Workload: The Wright-Pa	AL/CFHP	12 - 3
Heather E Castellano	East Central High School , San Antonio , TX The Directive Role of Statistics in Medicine	AL/AOCR	12 - 4
Christopher J Chadwell	James Madison High School , San antonio , TX A Pascal Program for a PC-Based Data Acquisition S	AL/CFT	12 - 5
Eleanore J Chuang	Beavercreek High School , Beavercreek , OH Evaluation of Head Scans From the HGU-53/P Helmet	AL/CFHD	12 - 6
Clayton J Ciomperlik	East Central High School , San Antonio , TX Concentrations of Radionuclides	AL/OEB	12 - 7
Kara L Ciomperlik	East Central High School , San Antonio , TX Analysis of Various Samples for the Presence of Me	AL/OEA	12 - 8
Joseph A Croswell	A Crawford Mosley High , Lynn Haven , FL Network Applications	AL/EQ	12 - 9
Timothy O Dickson	Rutherford High School , Springfield , FL Study, Design, and Modification of the Dynamic Con	AL/EQP	12 - 10
Maureen D Finke	New Braunfels High School , New Braunfels , TX An Optimization Study on a 99% Purity Molecular Si	AL/CFTS	12 - 11

Author	University/Institution Report Title	Armstrong Laboratory Directorate	Vol-Page
Angela D Foth	A. Crawford Mosley High School , Lynn Haven , FL Physical and Chemical Characterization of Columbus	AL/EQC	12 - 12
Andrea L Freeman	Judson High School , Converse , TX A Study of the Mortality Rate of the TEst Organisms	AL/OEM	12 - 13
Jeffrey P Gavornik	Roosevelt High School , San Antonio , TX A Study on the Effects of Chronic Intermittent Exp	AL/CFT	12 - 14
Mark W Giles	Bay High School , Panama City , FL Environmental Restoration Technologies Research	AL/EQW	12 - 15
Michael L Gunzburger	Kettering High School , Kettering , OH Programming Filtering Routines in the C Programmin	AL/CFBV	12 - 16
Brian C Harmon	A. Crawford Mosley High , Lynn Haven , FL A Study of the Nitrobenzene Reductase and its Reac	AL/EQC	12 - 17
Wesley R Hunt	James Madison High School , San Antonio , TX The Knowledge Survey and Assessment (KSA) Project	AL/HRM	12 - 18
Karen M Johnson	James Madison High School , San Antonio , TX Hyperbaric Medicine	AL/AOH	12 - 19
Damian A Kemper	Winston Churchill High School , San Antonio , TX Perception of the Spoken Stimuli in the S.C.O.N.E.	AL/AOCF	12 - 20
Nathan R Large	Northwestern High School , Springfield , OH A Paradigm for Studying Mutually Advantageous Trad	AL/CFHD	12 - 21
Trang D Le	Brackenridge High School , San Antonio , TX The Spacecraft Charging and Discharging Problem	AL/OEM	12 - 22

SRP Final Report Table of Contents

Author	University/Institution Report Title	Armstrong Laboratory Directorate	Vol-Page
Adriana Y Lopez	East Central High School , San Antonio , TX An Analysis of Oil/Grease in Water and Soil	AL/OEA	12 - 23
Steve J Mattingley	Mosley High School , Lynn Haven , FL A Study of the Practicality of an Automated Airfie	AL/EQ	12 - 24
Elizabeth A McKinley	Tecumseh High School , New Carlisle , OH Digitizing of Technical Illustrations	AL/HRG	12 - 25
David P McManamon	Carroll High School , Dayton , OH REPORT NOT AVAILABLE AT PRESS TIME	AL/CFBA	12 - 26
Amanda L Olson	Rutherford High School , Panama City , FL Physical and Chemical Characterization of Columbus	AL/EQC	12 - 27
Christopher S Protz	A. Crawford Mosley High School , Lynn Haven , FL Network Considerations	AL/EQP	12 - 28
Sarah E Schanding	East Central High School , San Antonio , TX REPORT NOT AVAILABLE AT PRESS TIME	AL/CFTF	12 - 29
Rebecca J Scheel	James Madison High School , San Antonio , TX The Learning of Hyperbaric Medicine	AL/AOH	12 - 30
Tina K Schuster	Southwest High School , San Antonio , TX The Determination of Lead in Paint Chips	AL/OEA	12 - 31
Kirk M Sexton	Northside Hlth Careers HS , San Antonio , TX Predicting Performance in Real-Time Tasks	AL/HRM	12 - 32
Ryan Q Simon	Beavercreek High School , Beavercreek , OH The Combustion of Advanced Composite Materials	AL/OET	12 - 33

SRP Final Report Table of Contents

Author	University/Institution Report Title	Armstrong Laboratory Directorate	Vol-Page
Kenneth B Spears	Highlands High School , San Antonio , TX Molecular Modeling and Editing of Dalm Halides	AL/OER	12 - 34
Courtney A Sprague	Southwest High School , San Antonio , TX A Study of the Visual Tests Performed on Air Force	AL/AOCO	12 - 35
Jonathan S Vinarskai	Castle Hls First Baptist Schoo , San Antonio , TX Which is a Better Sleep Scoring Device for Operati	AL/CFTO	12 - 36
Zac J Westbrook	Somerset High School , Somerset , TX The Effectiveness of Hyperbaric Oxygen Therapy in	AL/AOH	12 - 37
Thomas E Whalen	Carroll High School , Dayton , OH Utility of Internet Based Information Systems in A	AL/CFBE	12 - 38

SRP Final Report Table of Contents

Author	University/Institution Report Title	Phillips Laboratory Directorate	Vol-Page
Christopher D Amos	Desert High School , Edwards , CA Thermal Analysis of HADN and S-HAN-5	PL/RKAP	13 - 1
Rhianna S DaSalla	West Mesa High School , Albuquerque , NM Reflected Laser Communication Systems	PL/SXO	13 - 2
Alexander E Duff	La Cueva High School , Albuquerque , NM Construction and Testing of a Dual Photodiode Rece	PL/LIMI	13 - 3
Bridget C Engelhardt	Paraclete High School , Lancaster , CA A Study of Liner Compositions for Solution Propell	PL/RKAP	13 - 4
Daniel C Ghiglia	Sandia Prep High School , Albuquerque , NM The Construction of a Model Solar Powered Car	PL/VTPC	13 - 5
Tad Goetz	Sandia Preparatory High School , Albuquerque , NM Theoretical Study of Radiation and Heating Effects	PL/VTET	13 - 6
DeLesley S Hutchins	Albuquerque High School , Albuquerque , NM Programming Data Classification Procedures, Time M	PL/LIAE	13 - 7
Caroline H Lee	Lexington Sr. High School , Lexington , MA The Spacecraft Charging and Discharging Problem	PL/WSSI	13 - 8
David P Mirabal	West Mesa High School , Albuquerque , NM High Altitude Ballon Capabilities and Options	PL/SXO	13 - 9
Nicholas P Mitchell	Belen High School , Belen , NM Development of the PICLL (Particle in Cell Linked	PL/WSP	13 - 10
Julie A Niemeyer	Valley High School , Albuquerque , NM Nickel-Cadmium Batteries	PL/VTSI	13 - 11

Author	University/Institution Report Title	Phillips Laboratory Directorate	Vol-Page
Krista M Nuttall	La Cueva High School , Albuquerque , NM The Characterization of an Atmospheric Turbulence	PL/LIMI	13 - 12
Matthew J Pepper	St. Pius X High School , Albuquerque , NM The PSPH Computer Code an the WSCD Reference Datab	PL/WSCE	13 - 13
Jeremy G Pepper	St. Pius X High School , Albuquerque , NM A Study of the CIV Phenomenon and the Secondary an	PL/WSCD	13 - 14
Paul A Rodriguez	Santa Fe High School , Santa Fe , NM Using Image Processing Programs to Aid Space to Gr	PL/LIMI	13 - 15
Alok J Saldanha	Philips Academy , Andover , MA REPORT NOT AVAILABLE AT PRESS TIME	PL/GPSG	13 - 16
David M Schindler	Los Lunas High School , Los Lunas , NM Projects in the Nonlinear Optics Branch of the Phi	PL/LIDN	13 - 17
Min Shao	Arlington High School , Arlington , MA A Study of the Ionsphere	PL/GPIA	13 - 18
Raul Torrez	Sandia Preparatory School , Albuquerque , NM A Study of Infrared Devices and RAdiometric Measur	PL/VTRP	13 - 19
Christian G Warden	Rosamond High School , Rosamond , CA Introduction to Electric Propulsion	PL/RKCO	13 - 20

SRP Final Report Table of Contents

Author	University/Institution Report Title	Rome Laboratory Directorate	Vol-Page
Thomas J Angell	Camden Central High , Camden , NY A Comparison Between Relational Databases and Obje	RL/C3AA	14 - 1
Jonathan C Bakert	Sauquoit Valley Central High S , Sauquoit , NY C Programming for Digital Analysis and the Unix Op	RL/ERDA	14 - 2
Craig M Belusar	Oneida High School , Oneida , NY A Study in the Development of Specialized Software	RL/IRAP	14 - 3
Shawn H Bisgrove	Rome Free Academy , Rome , NY Arc-Second Raster Chart/Map Digitized Raster Grap	RL/IRRP	14 - 4
Stacy R Fitzsimmons	Vernon Verona Sherrill Cen Sch , Verona , NY An Implementation of the Multiple Signal Classific	RL/IRAA	14 - 5
David W Gurecki	Rome Catholic High School , Rome , NY The Information Superhighway: Still Under Constr	RL/C3B	14 - 6
Eric J Hayduk	Rome Catholic High School , Rome , NY Developing a Software Environment for a High Perfo	RL/OCTS	14 - 7
Justin D O'Brien	Bishop Guertin High School , Nashua , NH REPORT NOT AVAILABLE AT PRESS TIME	RL/ERMH	14 - 8
Michael J Panara	Rome Free Academy , Rome , NY Multi-Media-Creation and Uses (Using the MacroMind	RL/C3CA	14 - 9
Anne E Pletl	Notre Dame , Utica , NY Study of Global Hypermedia Networks	RL/C3BC	14 - 10
Richard A Schneible	Trivium School , Lancaster , MA Developing a Software Environment for a High Perfo	RL/OCTS	14 - 11

SRP Final Report Table of Contents

Author	University/Institution Report Title	Rome Laboratory Directorate	Vol-Page
Nathan B Terry	Clinton High School , Clinton , NY ADESH as a Sample Generator for mdem	RL/ERDR	14 - 12
Brian P Testa	Oxford Road , New Hartford , NY The Physical Significance of the Eigenvalues in Ad	RL/OCTS	14 - 13

SRP Final Report Table of Contents

Author	University/Institution Report Title	Wright Laboratory Directorate	Vol-Page
Christine M Baker	Norhmont High School , Cayton , OH Thermal Stresses in Composite Materials	WL/FIOP	15 - 1
Jennifer Bautista	Fort Walton Beach High , Fort Walton Beach , FL Analysis of a Three-Penetrator Concrete Penetratio	WL/MNOE	15 - 2
Jessica M Behm	Kettering Fairmont High School , Kettering , OH A Study of Silk Coatings on Thin Films	WL/MLPJ	15 - 3
Tim B Booher	Tippecanoe High School , Tipp City , OH Analysis of Spectrum Loading of SCS-6/Timetal 21s	WL/MLLM	15 - 4
Kim Cabral	Choctawhatchee High School , Ft. Walton Beach , FL Chemical Decomposition Using Non-Thermal Discharge	WL/MNOE	15 - 5
Robyn M Carley	Ft. Walton Beach High School , Ft. Walton Beach , FL Accuracy Verification Exercise for the Composite H	WL/MNOE	15 - 6
Jason P Carranza	Chaminade-Julienne High School , Dayton , OH The Adams Project	WL/AAAF-	15 - 7
George P Choung	Beavercreek High School , Beavercreek , OH Development of Astros, Version II for a Personal C	WL/FIOP	15 - 8
Nick D DeBrosse	Kettering Fairmont High School , Kettering , OH Advanced Gas Turbine Engine Compressor Design	WL/POTF	15 - 9
Nancy H Deibler	Choctawhatchee High School , Ft. Walton Beach , FL Characterization of Core Soil Samples and Plants F	WL/MNOE	15 - 10
Timothy G Donohue	Carroll High School , Dayton , OH The Building of Computer Programs and Inexpensive	WL/FIOP	15 - 11

SRP Final Report Table of Contents

Author	University/Institution Report Title	Wright Laboratory Directorate	Vol-Page
Michael J Dooley	Niceville High School , Niceville , FL Investigation of Programming and UNIX Applications	WL/MNOE _____	15 - 12
Ajay Goel	Centerville High School , Centerville , OH A Study of Polymer Dispersed Liquid Crystals	WL/MLPJ _____	15 - 13
Christie Gooden	Fort Walton Beach High School , Fort Walton Beach , FL Automated Integration of LADAR Imagery and TIFF St	WL/MNOE _____	15 - 14
Gary L Grogg	Carroll High School , Dayton , OH Heat Pipe Compatibility with Aircraft	WL/POOS _____	15 - 15
Matthew T Gudorf	Carroll High School , Dayton , OH The Analog Systems in Test Cell 22	WL/POPT _____	15 - 16
Brian J Guilfoos	Kettering High School , Kettering , OH CAD: A Testing of the Effectiveness of Process De	WL/MLIM _____	15 - 17
Douglas J Heil	Vandalia-Butler , Vandalia , OH Projects in Pattern Theory	WL/AART- _____	15 - 18
Laura L Hemmer	Choctawhatchee High School , Ft. Walton Beach , FL High Surface Area Conductive Polymer Films Using A	WL/MNOE _____	15 - 19
David B Hernandez	Freeport High School , Freeport , FL Preliminary Study for Application of IRMA Syntheti	WL/MNOE _____	15 - 20
Melanie L Hodges	W.Carrollton Sr. High School , West Carrollton , OH Parallel Gaseous Fuel Injecton into a Mach 2 Frees	WL/POPT _____	15 - 21
Venessa L Hurst	Walton Senior High School , DeFuniak Springs , FL Fluorodenitration of Aromatic Substrates	WL/MNOE _____	15 - 22

SRP Final Report Table of Contents

Author	University/Institution Report Title	Wright Laboratory Directorate	Vol-Page
Ryan A Jasper	Carroll High School , Dayton , OH Experiments in Fuel Research	WL/POSF _____	15- 23
Mark E Jeffcoat	Choctawhatchee High School , Ft. Walton Beach , FL Segmentation of an M-60 Tank from a High-Clutter B	WL/MNOE _____	15- 24
Andrew J Konicki	Kettering Fairmont High School , Kettering , OH Carbon-Carbon Structures Test	WL/FIOP _____	15- 25
Barry Kress	Niceville High , Niceville , FL The Effectiveness and Accuracy of Cadra Software	WL/MNOE _____	15- 26
Sandra R McPherson	Bishop Brossart High School , Alexandria , KY A Study of KTA	WL/MLPO _____	15- 27
Benjamin J Merrill	Bellbrook High School , Bellbrook , OH Visual Instrumentation Development	WL/FIOP _____	15- 28
Gary W Midkiff	Kettering Fairmont High School , Kettering , OH Porting Spice 2G.6 to UNIX	WL/ELED _____	15- 29
Karthik Natarajan	Beavercreek High School , Beavercreek , OH A Study of the Organic Reactions of Phthalocyanine	WL/MLPJ _____	15- 30
Christina L Noll	Tritwood-Madison High , Trotwood , OH A Study of the Viscosity of Lubricating Oils	WL/POSL _____	15- 31
Joanna E Odella	Kettering Fairmont High School , Kettering , OH Starting Here and Going Beyond	WL/AAAI- _____	15- 32
Alexander Penn	Niceville High School , Niceville , FL Design and Construction of a Fluorescence	WL/MNOE _____	15- 33

Author	University/Institution Report Title	Wright Laboratory Directorate	Vol-Page
Kyle Perry	Crestview High School , Crestview , FL Validation of Synthetic Imagery	WL/MNOE _____	15 - 34
Daniel R Pfunder	Centerville High School , Centerville , OH Integrated Generator Technology	WL/POOS _____	15 - 35
Mary Pletcher	Niceville High School , Niceville , FL Chemcial Characteristics of the Rocky Creek System	WL/MNOE _____	15 - 36
Scott E Sadowski	Centerville High School , Centerville , OH PAC vs. Area Methods of Determining "Learnability"	WL/AART- _____	15 - 37
Raul H Sanchez	Centerville High School , Centerville , OH Quantum Well Infrared Detector Research	WL/ELOD _____	15 - 38
Jill M Schlotterbeck	Kettering Fairmont High School , Kettering , OH A Study of Single Tube Catalyzed Heat Exchange	WL/POPT _____	15 - 39
Robert J Skebo	Beavercreek High School , Beavercreek , OH The Effect of Humidity on Friction and Wear for M5	WL/MLBT _____	15 - 40
Jennifer A Starr	Trotwood Madison Sr. High Scho , Trotwood , OH My Introduction to the Internet	WL/AAAF- _____	15 - 41
Todd D Stockert	Centerville High School , Centerville , OH The Effect of Temperature Upon Ho:YA1O3 Fluorescen	WL/ELOS _____	15 - 42
David B Storch	Beavercreek High School , Beavercreek , OH High Temperature/High Speed Laser Project	WL/ELR _____	15 - 43
Christopher J Sutton	Jefferson High School , Dayton , OH Dynamic Testing of Composites	WL/FIVS _____	15 - 44

SRP Final Report Table of Contents

Author	University/Institution Report Title	Wright Laboratory Directorate	Vol-Page
Thomas R Sutton	Sauquoit Valley High , Sauquoit , NY Dynamic Testing of Composites	WL/FIVS	15 - 45
Randy Thomson	Choctawhatchee High , Fort Walton Beach , FL Development and Testing of a Two-Dimensional Finit	WL/MNOE	15 - 46
John W Vest	Niceville High School , Niceville , FL Characterization of Optical Filters Built Using Sy	WL/MNOE	15 - 47
MR Jon R Ward	Walton High School , DeFuniak Springs , FL Data Acquisition, Reduction, and Storage Using Lab	WL/MNOE	15 - 48
Jeffrey D Warren	Fairborn High School , Fairborn , OH Computer Resource Team	WL/FIOP	15 - 49
Joshua A Weaver	Niceville High School , Niceville , FL Moments and Other PC Utilities	WL/MNOE	15 - 50
Gerad M Welch	Beavercreek High School , Beavercreek , OH Software Assisted Component Testing for the Antenn	WL/AAAI-	15 - 51
Gabrielle W WhiteWolf	Choctawhatchee High Schoo , Chcotawhatchee , FL Laser Speckle MTF Test Automation and Characteriza	WL/MNOE	15 - 52

Author	University/Institution Report Title	Arnold Engineering Development Center Directorate	Vol-Page
Ryan B Bond	Tullahoma High School , Tullahoma , TN Modeling Engine Test Facility Cells in Vissim	Sverdrup	16 - 1
Robert B Cassady	Coffee County Cen High School , Manchester , TN Mach-Flow Angularity Probe Calibration	Calspan	16 - 2
Thomas L Clouse	Coffee County Central HS , Manchester , TN Workstation Inventory Control Program	Sverdrup	16 - 3
Michael L Fann	Tullahoma High School , Tullahoma , TN The Conversion of Millivolts Measured from Thermoc	Sverdrup	16 - 4
Derek E Geeting	Shelbyville Central High , Shelbyville , TN Lighting Calculation Study and Software Evaluation	SSI	16 - 5
Jennifer A Groff	Franklin County Sr High School , Winchester , TN The Use of Labview for Serial Data Transmission	Sverdrup	16 - 6
James J Lemmons	Coffee County Central HS , Manchester , TN Out of Band Filter Calibration Technology Project	Bionetics	16 - 7
Lana L Matthews	Coffee County Central HS , Manchester , TN A Study of Hydrocarbon Combustion: Stoichiometry	Sverdrup	16 - 8
Steve G Pugh	Shelbyville Central HS , Shelbyville , TN An Analytic Capability for Predicting Sability of	Sverdrup	16 - 9
Kristopher S Ray	Shelbyville Central High Schoo , Shelbyville , TN Power Systems Analysis	SSI	16 - 10

1. INTRODUCTION

The Summer Research Program (SRP), sponsored by the Air Force Office of Scientific Research (AFOSR), offers paid opportunities for university faculty, graduate students, and high school students to conduct research in U.S. Air Force research laboratories nationwide during the summer.

Introduced by AFOSR in 1978, this innovative program is based on the concept of teaming academic researchers with Air Force scientists in the same disciplines using laboratory facilities and equipment not often available at associates' institutions.

AFOSR also offers its research associates an opportunity, under the Summer Research Extension Program (SREP), to continue their AFOSR-sponsored research at their home institutions through the award of research grants. In 1994 the maximum amount of each grant was increased from \$20,000 to \$25,000, and the number of AFOSR-sponsored grants decreased from 75 to 60. A separate annual report is compiled on the SREP.

The Summer Faculty Research Program (SFRP) is open annually to approximately 150 faculty members with at least two years of teaching and/or research experience in accredited U.S. colleges, universities, or technical institutions. SFRP associates must be either U.S. citizens or permanent residents.

The Graduate Student Research Program (GSRP) is open annually to approximately 100 graduate students holding a bachelor's or a master's degree; GSRP associates must be U.S. citizens enrolled full time at an accredited institution.

The High School Apprentice Program (HSAP) annually selects about 125 high school students located within a twenty mile commuting distance of participating Air Force laboratories.

The numbers of projected summer research participants in each of the three categories are usually increased through direct sponsorship by participating laboratories.

AFOSR's SRP has well served its objectives of building critical links between Air Force research laboratories and the academic community, opening avenues of communications and forging new research relationships between Air Force and academic technical experts in areas of national interest; and strengthening the nation's efforts to sustain careers in science and engineering. The success of the SRP can be gauged from its growth from inception (see Table 1) and from the favorable responses the 1994 participants expressed in end-of-tour SRP evaluations (Appendix B).

AFOSR contracts for administration of the SRP by civilian contractors. The contract was first awarded to Research & Development Laboratories (RDL) in September 1990. After completion of the 1990 contract, RDL won the recompetition for the basic year and four 1-year options.

2. PARTICIPATION IN THE SUMMER RESEARCH PROGRAM

The SRP began with faculty associates in 1979; graduate students were added in 1982 and high school students in 1986. The following table shows the number of associates in the program each year.

Table 1: SRP Participation, by Year

YEAR	Number of Participants			TOTAL
	SFRP	GSRP	HSAP	
1979	70			70
1980	87			87
1981	87			87
1982	91	17		108
1983	101	53		154
1984	152	84		236
1985	154	92		246
1986	158	100	42	300
1987	159	101	73	333
1988	153	107	101	361
1989	168	102	103	373
1990	165	121	132	418
1991	170	142	132	444
1992	185	121	159	464
1993	187	117	136	440
1994	192	117	133	442

Beginning in 1993, due to budget cuts, some of the laboratories weren't able to afford to fund as many associates as in previous years; in one case a laboratory did not fund any additional associates. However, the table shows that, overall, the number of participating associates increased this year because two laboratories funded more associates than they had in previous years.

3. RECRUITING AND SELECTION

The SRP is conducted on a nationally advertised and competitive-selection basis. The advertising for faculty and graduate students consisted primarily of the mailing of 8,000 44-page SRP brochures to chairpersons of departments relevant to AFOSR research and to administrators of grants in accredited universities, colleges, and technical institutions. Historically Black Colleges and Universities (HBCUs) and Minority Institutions (MIs) were included. Brochures also went to all participating USAF laboratories, the previous year's participants, and numerous (over 600 annually) individual requesters.

Due to a delay in awarding the new contract, RDL was not able to place advertisements in any of the following publications in which the SRP is normally advertised: *Black Issues in Higher Education*, *Chemical & Engineering News*, *IEEE Spectrum* and *Physics Today*.

High school applicants can participate only in laboratories located no more than 20 miles from their residence. Tailored brochures on the HSAP were sent to the head counselors of 180 high schools in the vicinity of participating laboratories, with instructions for publicizing the program in their schools. High school students selected to serve at Wright Laboratory's Armament Directorate (Eglin Air Force Base, Florida) serve eleven weeks as opposed to the eight weeks normally worked by high school students at all other participating laboratories.

Each SFRP or GSRP applicant is given a first, second, and third choice of laboratory. High school students who have more than one laboratory or directorate near their homes are also given first, second, and third choices.

Laboratories make their selections and prioritize their nominees. AFOSR then determines the number to be funded at each laboratory and approves laboratories' selections.

Subsequently, laboratories use their own funds to sponsor additional candidates. Some selectees do not accept the appointment, so alternate candidates are chosen. This multi-step selection procedure results in some candidates being notified of their acceptance after scheduled deadlines. The total applicants and participants for 1994 are shown in this table.

Table 2: 1994 Applicants and Participants

PARTICIPANT CATEGORY	TOTAL APPLICANTS	SELECTEES	DECLINING SELECTEES
SFRP	600	192	30
(HBCU/MI)	(90)	(16)	(7)
GSRP	322	117	11
(HBCU/MI)	(11)	(6)	(0)
HSAP	562	133	14
TOTAL	1484	442	55

4. SITE VISITS

During June and July of 1994, representatives of both AFOSR/NI and RDL visited each participating laboratory to provide briefings, answer questions, and resolve problems for both laboratory personnel and participants. The objective was to ensure that the SRP would be as constructive as possible for all participants. Both SRP participants and RDL representatives found these visits beneficial. At many of the laboratories, this was the only opportunity for all participants to meet at one time to share their experiences and exchange ideas.

5. HISTORICALLY BLACK COLLEGES AND UNIVERSITIES AND MINORITY INSTITUTIONS (HBCU/MIs)

In previous years, an RDL program representative visited from seven to ten different HBCU/MIs to promote interest in the SRP among the faculty and graduate students. Due to the late contract award date (January 1994) no time was available to visit HBCU/MIs this past year.

In addition to RDL's special recruiting efforts, AFOSR attempts each year to obtain additional funding or use leftover funding from cancellations the past year to fund HBCU/MI associates. This year, seven HBCU/MI SFRPs declined after they were selected. The following table records HBCU/MI participation in this program.

Table 3: SRP HBCU/MI Participation, by Year

YEAR	SFRP		GSRP	
	Applicants	Participants	Applicants	Participants
1985	76	23	15	11
1986	70	18	20	10
1987	82	32	32	10
1988	53	17	23	14
1989	39	15	13	4
1990	43	14	17	3
1991	42	13	8	5
1992	70	13	9	5
1993	60	13	6	2
1994	90	16	11	6

6. SRP FUNDING SOURCES

Funding sources for the 1994 SRP were the AFOSR-provided slots for the basic contract and laboratory funds. Funding sources by category for the 1994 SRP selected participants are shown here.

Table 4: 1994 SRP Associate Funding

FUNDING CATEGORY	SFRP	GSRP	HSAP
AFOSR Basic Allocation Funds	150	98 ^{*1}	121 ^{*2}
USAF Laboratory Funds	37	19	12
HBCU/MI By AFOSR (Using Procured Addn'l Funds)	5	0	0
TOTAL	192	117	133

*1 - 100 were selected, but two canceled too late to be replaced.

*2 - 125 were selected, but four canceled too late to be replaced.

7. COMPENSATION FOR PARTICIPANTS

Compensation for SRP participants, per five-day work week, is shown in this table.

Table 5: 1994 SRP Associate Compensation

PARTICIPANT CATEGORY	1991	1992	1993	1994
Faculty Members	\$690	\$718	\$740	\$740
Graduate Student (Master's Degree)	\$425	\$442	\$455	\$455
Graduate Student (Bachelor's Degree)	\$365	\$380	\$391	\$391
High School Student (First Year)	\$200	\$200	\$200	\$200
High School Student (Subsequent Years)	\$240	\$240	\$240	\$240

APPENDIX A -- PROGRAM STATISTICAL SUMMARY

A. Colleges/Universities Represented

Selected SFRP and GSRP associates represent 158 different colleges, universities, and institutions.

B. States Represented

SFRP - Applicants came from 46 states plus Washington D.C. and Puerto Rico. Selectees represent 40 states.

GSRP - Applicants came from 46 states and Puerto Rico. Selectees represent 34 states.

HSAP - Applicants came from fifteen states. Selectees represent ten states.

C. Academic Disciplines Represented

The academic disciplines of the combined 192 SFRP associates are as follows:

Electrical Engineering	22.4%
Mechanical Engineering	14.0%
Physics: General, Nuclear & Plasma	12.2%
Chemistry & Chemical Engineering	11.2%
Mathematics & Statistics	8.1%
Psychology	7.0%
Computer Science	6.4%
Aerospace & Aeronautical Engineering	4.8%
Engineering Science	2.7%
Biology & Inorganic Chemistry	2.2%
Physics: Electro-Optics & Photonics	2.2%
Communication	1.6%
Industrial & Civil Engineering	1.6%
Physiology	1.1%
Polymer Science	1.1%
Education	0.5%
Pharmaceutics	0.5%
Veterinary Medicine	0.5%
TOTAL	100%

Table A-1. Total Participants

Number of Participants	
SFRP	192
GSRP	117
HSAP	133
TOTAL	442

Table A-2. Degrees Represented

Degrees Represented			
	SFRP	GSRP	TOTAL
Doctoral	189	0	189
Master's	3	47	50
Bachelor's	0	70	70
TOTAL	192	117	309

Table A-3. SFRP Academic Titles

Academic Titles	
Assistant Professor	74
Associate Professor	63
Professor	44
Instructor	5
Chairman	1
Visiting Professor	1
Visiting Assoc. Prof.	1
Research Associate	3
TOTAL	192

Table A-4. Source of Learning About SRP

SOURCE	SFRP		GSRP	
	Applicants	Selectees	Applicants	Selectees
Applied/participated in prior years	26 %	37 %	10 %	13 %
Colleague familiar with SRP	19 %	17 %	12 %	12 %
Brochure mailed to institution	32 %	18 %	19 %	12 %
Contact with Air Force laboratory	15 %	24 %	9 %	12 %
Faculty Advisor (GSRPs Only)	--	--	39 %	43 %
Other source	8 %	4 %	11 %	8 %
TOTAL	100 %	100 %	100 %	100 %

Table A-5. Ethnic Background of Applicants and Selectees

	SFRP		GSRP		HSAP	
	Applicants	Selectees	Applicants	Selectees	Applicants	Selectees
American Indian or Native Alaskan	0.2 %	0 %	1 %	0 %	0.4 %	0 %
Asian/Pacific Islander	30 %	20 %	6 %	8 %	7 %	10 %
Black	4 %	1.5 %	3 %	3 %	7 %	2 %
Hispanic	3 %	1.9 %	4 %	4.5 %	11 %	8 %
Caucasian	51 %	63 %	77 %	77 %	70 %	75 %
Preferred not to answer	12 %	14 %	9 %	7 %	4 %	5 %
TOTAL	100 %	100 %	100 %	100 %	99 %	100 %

Table A-6. Percentages of Selectees receiving their 1st, 2nd, or 3rd Choices of Directorate

	1st Choice	2nd Choice	3rd Choice	Other Than Their Choice
SFRP	70 %	7 %	3 %	20 %
GSRP	76 %	2 %	2 %	20 %

APPENDIX B – SRP EVALUATION RESPONSES

1. OVERVIEW

Evaluations were completed and returned to RDL by four groups at the completion of the SRP. The number of respondents in each group is shown below.

Table B-1. Total SRP Evaluations Received

Evaluation Group	Responses
SFRP & GSRPs	275
HSAPs	116
USAF Laboratory Focal Points	109
USAF Laboratory HSAP Mentors	54

All groups indicate near-unanimous enthusiasm for the SRP experience.

Typical comments from 1994 SRP associates are:

"[The SRP was an] excellent opportunity to work in state-of-the-art facility with top-notch people."

"[The SRP experience] enabled exposure to interesting scientific application problems; enhancement of knowledge and insight into 'real-world' problems."

"[The SRP] was a great opportunity for resourceful and independent faculty [members] from small colleges to obtain research credentials."

"The laboratory personnel I worked with are tremendous, both personally and scientifically. I cannot emphasize how wonderful they are."

"The one-on-one relationship with my mentor and the hands on research experience improved [my] understanding of physics in addition to improving my library research skills. Very valuable for [both] college and career!"

Typical comments from laboratory focal points and mentors are:

"This program [AFOSR - SFRP] has been a 'God Send' for us. Ties established with summer faculty have proven invaluable."

"Program was excellent from our perspective. So much was accomplished that new options became viable "

"This program managed to get around most of the red tape and 'BS' associated with most Air Force programs. Good Job!"

"Great program for high school students to be introduced to the research environment. Highly educational for others [at laboratory]."

"This is an excellent program to introduce students to technology and give them a feel for [science/engineering] career fields. I view any return benefit to the government to be 'icing on the cake' and have usually benefitted."

The summarized recommendations for program improvement from both associates and laboratory personnel are listed below (Note: basically the same as in previous years.)

- A. Better preparation on the labs' part prior to associates' arrival (i.e., office space, computer assets, clearly defined scope of work).
- B. Laboratory sponsor seminar presentations of work conducted by associates, and/or organized social functions for associates to collectively meet and share SRP experiences.
- C. Laboratory focal points collectively suggest more AFOSR allocated associate positions, so that more people may share in the experience.
- D. Associates collectively suggest higher stipends for SRP associates.
- E. Both HSAP Air Force laboratory mentors and associates would like the summer tour extended from the current 8 weeks to either 10 or 11 weeks; the groups state it takes 4-6 weeks just to get high school students up-to-speed on what's going on at laboratory. (Note: this same argument was used to raise the faculty and graduate student participation time a few years ago.)

2. 1994 USAF LABORATORY FOCAL POINT (LFP) EVALUATION RESPONSES

The summarized results listed below are from the 109 LFP evaluations received.

1. LFP evaluations received and associate preferences:

Table B-2. Air Force LFP Evaluation Responses (By Type)

		How Many Associates Would You Prefer To Get ?								(% Response)			
Lab	Evals Recv'd	SFRP				GSRP (w/Univ Professor)				GSRP (w/o Univ Professor)			
		0	1	2	3+	0	1	2	3+	0	1	2	3+
AEDC	10	30	50	0	20	50	40	0	10	40	60	0	0
AL	44	34	50	6	9	54	34	12	0	56	31	12	0
FJSRL	3	33	33	33	0	67	33	0	0	33	67	0	0
PL	14	28	43	28	0	57	21	21	0	71	28	0	0
RL	3	33	67	0	0	67	0	33	0	100	0	0	0
WHMC	1	0	0	100	0	0	100	0	0	0	100	0	0
WL	46	15	61	24	0	56	30	13	0	76	17	6	0
Total	121	25%	43%	27%	4%	50%	37%	11%	1%	54%	43%	3%	0%

LFP Evaluation Summary. The summarized responses, by laboratory, are listed on the following page. LFPs were asked to rate the following questions on a scale from 1 (below average) to 5 (above average).

2. LFPs involved in SRP associate application evaluation process:
 - a. Time available for evaluation of applications:
 - b. Adequacy of applications for selection process:
3. Value of orientation trips:
4. Length of research tour:
5.
 - a. Benefits of associate's work to laboratory:
 - b. Benefits of associate's work to Air Force:
6.
 - a. Enhancement of research qualifications for LFP and staff:
 - b. Enhancement of research qualifications for SFRP associate:
 - c. Enhancement of research qualifications for GSRP associate:
7.
 - a. Enhancement of knowledge for LFP and staff:
 - b. Enhancement of knowledge for SFRP associate:
 - c. Enhancement of knowledge for GSRP associate:
8. Value of Air Force and university links:
9. Potential for future collaboration:
10.
 - a. Your working relationship with SFRP:
 - b. Your working relationship with GSRP:
11. Expenditure of your time worthwhile:

(Continued on next page)

12. Quality of program literature for associate:
13. a. Quality of RDL's communications with you:
 b. Quality of RDL's communications with associates:
14. Overall assessment of SRP:

Laboratory Focal Point Responses to above questions							
	<i>AEDC</i>	<i>AL</i>	<i>FJSRL</i>	<i>PL</i>	<i>RL</i>	<i>WHMC</i>	<i>WL</i>
<i># Evals Recv'd</i>	10	32	3	14	3	1	46
<i>Question #</i>							
2	90 %	62 %	100 %	64 %	100 %	100 %	83 %
2a	3.5	3.5	4.7	4.4	4.0	4.0	3.7
2b	4.0	3.8	4.0	4.3	4.3	4.0	3.9
3	4.2	3.6	4.3	3.8	4.7	4.0	4.0
4	3.8	3.9	4.0	4.2	4.3	NO ENTRY	4.0
5a	4.1	4.4	4.7	4.9	4.3	3.0	4.6
5b	4.0	4.2	4.7	4.7	4.3	3.0	4.5
6a	3.6	4.1	3.7	4.5	4.3	3.0	4.1
6b	3.6	4.0	4.0	4.4	4.7	3.0	4.2
6c	3.3	4.2	4.0	4.5	4.5	3.0	4.2
7a	3.9	4.3	4.0	4.6	4.0	3.0	4.2
7b	4.1	4.3	4.3	4.6	4.7	3.0	4.3
7c	3.3	4.1	4.5	4.5	4.5	5.0	4.3
8	4.2	4.3	5.0	4.9	4.3	5.0	4.7
9	3.8	4.1	4.7	5.0	4.7	5.0	4.6
10a	4.6	4.5	5.0	4.9	4.7	5.0	4.7
10b	4.3	4.2	5.0	4.3	5.0	5.0	4.5
11	4.1	4.5	4.3	4.9	4.7	4.0	4.4
12	4.1	3.9	4.0	4.4	4.7	3.0	4.1
13a	3.8	2.9	4.0	4.0	4.7	3.0	3.6
13b	3.8	2.9	4.0	4.3	4.7	3.0	3.8
14	4.5	4.4	5.0	4.9	4.7	4.0	4.5

3. 1994 SFRP & GSRP EVALUATION RESPONSES

The summarized results listed below are from the 275 SFRP/GSRP evaluations received.

Associates were asked to rate the following questions on a scale from
1 (below average) to 5 (above average)

1. The match between the laboratories research and your field:	4.6
2. Your working relationship with your LFP:	4.8
3. Enhancement of your academic qualifications:	4.4
4. Enhancement of your research qualifications:	4.5
5. Lab readiness for you: LFP, task, plan:	4.3
6. Lab readiness for you: equipment, supplies, facilities:	4.1
7. Lab resources:	4.3
8. Lab research and administrative support:	4.5
9. Adequacy of brochure and associate handbook:	4.3
10. RDL communications with you:	4.3
11. Overall payment procedures:	3.8
12. Overall assessment of the SRP:	4.7
13. a. Would you apply again?	Yes: 85 %
b. Will you continue this or related research?	Yes: 95 %
14. Was length of your tour satisfactory?	Yes: 86 %
15. Percentage of associates who engaged in:	
a. Seminar presentation:	52 %
b. Technical meetings:	32 %
c. Social functions:	03 %
d. Other	01 %

16. Percentage of associates who experienced difficulties in:

- | | |
|---------------------|------|
| a. Finding housing: | 12 % |
| b. Check Cashing: | 03 % |

17. Where did you stay during your SRP tour?

- | | |
|----------------------|------|
| a. At Home: | 20 % |
| b. With Friend: | 06 % |
| c. On Local Economy: | 47 % |
| d. Base Quarters: | 10 % |

THIS SECTION FACULTY ONLY:

18. Were graduate students working with you? Yes: 23 %

19. Would you bring graduate students next year? Yes: 56 %

20. Value of orientation visit:

- | | |
|-----------------|------|
| Essential: | 29 % |
| Convenient: | 20 % |
| Not Worth Cost: | 01 % |
| Not Used: | 34 % |

THIS SECTION GRADUATE STUDENTS ONLY:

21. Who did you work with:

- | | |
|-----------------------|------|
| University Professor: | 18 % |
| Laboratory Scientist: | 54 % |

4. 1994 USAF LABORATORY HSAP MENTOR EVALUATION RESPONSES

The summarized results listed below are from the 54 mentor evaluations received.

1. Mentor apprentice preferences:

Table B-3. Air Force Mentor Responses

		How Many Apprentices Would You Prefer To Get ?			
		<i>HSAP Apprentices Preferred</i>			
<i>Laboratory</i>	<i># Evals Recv'd</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3+</i>
AEDC	6	0	100	0	0
AL	17	29	47	6	18
PL	9	22	78	0	0
RL	4	25	75	0	0
WL	18	22	55	17	6
Total	54	20%	71%	5%	5%

Mentors were asked to rate the following questions on a scale from 1 (below average) to 5 (above average)

2. Mentors involved in SRP apprentice application evaluation process:
 - a. Time available for evaluation of applications:
 - b. Adequacy of applications for selection process:
3. Laboratory's preparation for apprentice:
4. Mentor's preparation for apprentice:
5. Length of research tour:
6. Benefits of apprentice's work to U.S. Air force:
7. Enhancement of academic qualifications for apprentice:
8. Enhancement of research skills for apprentice:
9. Value of U.S. Air Force/high school links:
10. Mentor's working relationship with apprentice:
11. Expenditure of mentor's time worthwhile:
12. Quality of program literature for apprentice:
13.
 - a. Quality of RDL's communications with mentors:
 - b. Quality of RDL's communication with apprentices:
14. Overall assessment of SRP:

	<i>AEDC</i>	<i>AL</i>	<i>PL</i>	<i>RL</i>	<i>WL</i>
<i># Evals Recv'd</i>	6	17	9	4	18
<i>Question #</i>					
2	100 %	76 %	56 %	75 %	61 %
2a	4.2	4.0	3.1	3.7	3.5
2b	4.0	4.5	4.0	4.0	3.8
3	4.3	3.8	3.9	3.8	3.8
4	4.5	3.7	3.4	4.2	3.9
5	3.5	4.1	3.1	3.7	3.6
6	4.3	3.9	4.0	4.0	4.2
7	4.0	4.4	4.3	4.2	3.9
8	4.7	4.4	4.4	4.2	4.0
9	4.7	4.2	3.7	4.5	4.0
10	4.7	4.5	4.4	4.5	4.2
11	4.8	4.3	4.0	4.5	4.1
12	4.2	4.1	4.1	4.8	3.4
13a	3.5	3.9	3.7	4.0	3.1
13b	4.0	4.1	3.4	4.0	3.5
14	4.3	4.5	3.8	4.5	4.1

5. 1994 HSAP EVALUATION RESPONSES

The summarized results listed below are from the 116 HSAP evaluations received.

HSAP apprentices were asked to rate the following questions on a scale from
1 (below average) to 5 (above average)

1. Match of lab research to you interest:	3.9
2. Apprentices working relationship with their mentor and other lab scientists:	4.6
3. Enhancement of your academic qualifications:	4.4
4. Enhancement of your research qualifications:	4.1
5. Lab readiness for you: mentor, task, work plan	3.7
6. Lab readiness for you: equipment supplies facilities	4.3
7. Lab resources: availability	4.3
8. Lab research and administrative support:	4.4
9. Adequacy of RDL's apprentice handbook and administrative materials:	4.0
10. Responsiveness of RDL's communications:	3.5
11. Overall payment procedures:	3.3
12. Overall assessment of SRP value to you:	4.5
13. Would you apply again next year?	Yes: 88%
14. Was length of SRP tour satisfactory?	Yes: 78%
15. Percentages of apprentices who engaged in:	
a. Seminar presentation:	48%
b. Technical meetings:	23%
c. Social functions:	18%

A COMPARISON BETWEEN RELATIONAL DATABASES
AND OBJECT-ORIENTED DATABASES

Thomas J. Angell

Camden High School
Oswego St.
Camden, NY 13316

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory at
Griffiss Air Force Base

August 1994

Table of Contents

Title page	pg. 1
Abstract	pg. 3
An Overview of Database and Database Management Systems	pg. 4
Differences Between Relational and Object-Oriented Databases	pg. 8
Conclusions	pg. 12
References	pg. 13

Abstract

This paper discusses several attributes common among database and database management systems. It also compares two types of database management systems, Relational and Object-Oriented. A database can provide users with a better way of finding or updating data, but it must correspond to the real world and be user friendly in order to readily accomplish this task.

AN OVERVIEW OF DATABASES AND DATABASE MANAGEMENT SYSTEMS

A database is a collection of related data. It is designed and built for a specific purpose, intended for a specific group of users (although) it should be considered that anyone could possibly be using it. It represents or models some aspect of the real world. A database management system (DBMS) is a collection of programs that allows users to create, maintain, and access a database.

What is to be expected of a DBMS? A DBMS must correspond to the real world in scale, magnitude, and complexity-but allow users to easily enter and retrieve information. The users have the most important role because they are the ones that will be actually using the DBMS. Their primary job is not to program the database but to retrieve information that is scattered throughout the organization to answer questions or make decisions. The user or potential user of a DBMS need not have a firm grasp of DBMSs concepts. The database should be set up so that the user can find the data he is looking for without having to know how to program the database. This makes for a more user friendly DBMS. The user needs to maintain the database by modifying or updating it. Data can be added, deleted, or changed. The user, directly or indirectly via application programs, operates with the DBMS in three modes, definition, access, and maintenance. The user defines the database by specifying the format of the data and the relationship among the data. The user accesses data by requesting the DBMS to retrieve portions of the database. The user must maintain the DBMS by modifying or updating the system with new data.

The DBMS needs to be able to change the data as the real world changes. The data should need only to be changed once without affecting the application programs because the data and programs need to be independent of each other. A DBMS must support a data structure that corresponds to the real world. The data structure must have entities and data items. Database technology allows an organization's data to be processed as an integrated whole. Integration of data offers several important advantages. Data is compatible with information processing systems so that new systems need not be created for "one of a kind" requests. In effect, more information can be obtained from existing data because when data is integrated, more derivations are possible.

In pre-database systems, each user had their own file on which to enter and update data. Since information changes, data needs to be updated. If different departments use "copies" or "duplicates" of the same data, then some departments might get updated and others may not. The most serious problem of data duplication is that it can lead to a lack of data integrity meaning data items representing some part of the real world then disagree with one another. A DBMS should eliminate data duplication by having the system purge out dated data to make sure that data integrity is not compromised. This saves file space and can reduce processing requirements but must support access to data by multiple users simultaneously since more departments will be using the database. Because data is shared there is a need for a data architecture that works for all users. Since data is shared, it does not belong to any one department and there is an increased need for control. When data is centralized in a database, one administrator can specialize in the maintenance of data. A

DBMS provides an enterprise for centralized control of its operational data and improves communication and integration between departments. A DBMS can lead to better data management and quicker response time between departments.

The DBMS needs to be able to protect itself from unauthorized users. The first level of security is physical access to the system and the next level is passwords on the system and in the program. Also the data can be written in encryption. Passwords are a common method that can be used to protect a database. They may be used for the entire database or any portion of the system. Passwords can also be used to secure types of functional activity such as a read password and a write password. Data encryption is another common technique of database security. Data is stored and transmitted in coded format. It can be very simple to program, for example, the binary representation of data can be modified by squaring it or adding a constant to it. Characters can be shifted or one alphabet can replace another. The coding scheme must be kept a secret and be changed periodically. Programs that do the encoding and decoding must be kept out of reach of would-be infiltrators by the use of passwords. Unfortunately, any security measure, no matter how complex, can be circumvented in some way. The best that can be done is to hassle the would-be infiltrator so that extensive risk and effort would be required to perform unauthorized processing.

Not only does the DBMS have to protect itself from unauthorized users but also authorized users. To protect data from corruption by authorized users the DBMS must provide integrity. If data is changed in one area of the database then that information is sent throughout the entire

database. Data definition is crucial to a DBMS so all users properly understand the data represented. Support for integrity requires support for semantic integrity constraints (rules). The data entered into the DBMS must conform to the constraints about format, use, and meaning. For example, a social security number must have a certain number of digits. Also, back-up and recovery allows users to roll-back and correct or eliminate erroneous data. The database should be downloaded onto a back-up so that it can be recovered from a variety of failures ranging from a program error to a systems crash.

DBMSs do provide many advantages, but there are also disadvantages. Although the number of updates are reduced, errors in that data are much more painful and harder to detect. Since DBMS software is complex and consumes resources, there is a need for skilled design, administration, and management personnel. Training for programmers and users may be necessary. Additional hardware might be needed because a DBMS will make an overloaded computer more overloaded and may be incompatible with other DBMS's. A DBMS will not solve coordination among users-that is a people problem. A DBMS can enforce a policy, but not create it. A database may be able to detect errors in data but only users can change them. A DBMS has an increased vulnerability to failure since a failure in one component of an integrated system can stop an entire database.

Database processing seems to meet a critical need; it can be a better way to provide information to users. Data is just recorded facts and figures; information is knowledge derived from data. By using a database, users can obtain information quicker and more completely than those without access.

DIFFERENCES BETWEEN RELATIONAL AND OBJECT-ORIENTED DATABASES

Object-Oriented databases management systems (OODBMSs) and Relational databases management systems (RDBMSs) have fundamentally, the same goals. As RDBMSs grew and became more powerful, it became increasingly difficult to ask for one single piece of data. This brought around the use of OODBMSs where an object, familiar to a group of users, is called up, not a specific piece of data.

A major difference between the two database management systems is their primary goals. In a RDBMSs, the primary goal is data independence. This means that the data representation on the computer is independent of the interface to the application. The most significant advantage of this is that the physical representation can be modified without affecting applications. The database can be completely reorganized at the physical level without recompiling programs or schema. Performance features such as indexes can be added or removed dynamically, tables can be partitioned across disk drives or compressed to reclaim unused space, and so on. None of this affects the applications, so they are maintained more easily. In RDBMSs each relation (table) is separate. Join commands relate the data that is in separate relations. An international standard language that is used to express the relational model is SQL (standard query language). It is a declarative, nonprocedural language that expresses the kind of data desired, not how to get it. This allows the database system to choose from alternative mechanisms and to obtain the data from physical realization of the database. The database management system can dynamically optimize

the way queries are executed, freeing the application programmer from this task. When relational data is designed, the process of normalization can be applied. Fully normalizing a data model produces a database in which redundancy has been eliminated. This has two advantages. First, it removes the possibility that portions of the database are out of sync due to redundant storage. Second, it usually minimizes the amount of data stored, reducing the overall size of the database and saving disk space. The relational model has three basic types of data: relation, tuple (row), and the attribute (column). The relational model also has three operations: select, join, and project. The relational model is easier to learn and use because most people are already familiar with the basic concepts, since they have already worked with tables, columns, and rows.

In OODBMSs, there primary goal is encapsulation. After traditional databases had been in use for some time, a need arose to associate certain procedures with the data and activate them when the data was accessed. Such procedures were used to help control the integrity of the data and its security. Sometimes, a value was computed rather than stored-a procedure referred to as putting "intelligence" in the database. OODBMSs take the idea of intelligent databases to its logical conclusion. No data is accessed except with the methods stored in the database. The data of all objects is therefore encapsulated. This means that the data can only be employed with the methods that are part of a class. Object-Oriented classes are intended to be reusable. Therefore, another goal of OODBMS is to achieve maximum reusability. Because of this, the class should be bug-free and should only be modified if absolutely necessary. Traditional database technology is designed to support processes that are subject to endless

modification. Therefore, data independence is necessary. The OODBMS supports classes, some of which never change. Change comes from interlinking classes in diverse ways. The data structures in the OODBMS should be tightly optimized to support the class in which they are encapsulated. The data for one object can be interlinked and stored together with another object, so that they can be accessed from one position of the access mechanism.

In OODBMSs, the data is active rather than passive. Requests cause objects, which are active, to execute their methods. Objects are composed of objects that in turn are composed of objects and so on. Because of this, the data structures for one object can become highly complex. Certain limited operations may be automatically triggered when data is used in a RDBMSs, but for all intensive purposes, data is passive. OODBMSs give higher performance than RDBMSs for applications with complex data.

OODBMSs outperform RDBMSs for applications with lots of data connectivity. They allow objects to refer directly to one another using soft pointers-making OODBMSs much faster in getting from object A to object B. OODBMSs make physical clustering more effective. Most database systems allow the developer to place related structures close to each other in disk storage. This dramatically reduces retrieval time for the related data, since all data is read with one disk read, instead of several. However, in a RDBMS, implementation objects get translated into tabular representations and typically get spread out over multiple tables. Thus, in a RDBMS these related rows must be clustered together, so that the whole object can be retrieved with one disk read. In an OODBMS, this is automatic. Furthermore, clustering related objects, such as all subparts of

an assembly, can dramatically affect the overall performance of an application. This is relatively straightforward in an OODBMS, since this represents the first level of clustering.

In contrast, physical clustering is typically impossible in an RDBMS, because it requires a second level of clustering—one level to cluster the rows representing individual objects and a second for the groups of rows that represent related objects. OODBMSs use diverse storage structures. Relations are one of the many data structures that can be used along with BLOBs (binary large objects). BLOBs are used for sound, images, video, and large unstructured bit streams. In RDBMSs, data that cannot be easily expressed in tabular form is difficult to store and access efficiently. For example, multimedia applications require the storage of large data streams representing digitized video and audio data. CAD (Computer Assisted Design) applications often require the storage of large numbers of very small objects, such as the point defining the geometry of a mechanical part. Neither is well-suited to representation in a table. Therefore, RDBMSs cannot provide efficient storage management for these application areas. The storage model for OODBMSs is unlimited, since the systems is, by nature, extensible. OODBMSs can provide different storage mechanisms for different kinds of data. As a result, they have proven very effective in supporting both multimedia and CAD applications.

Conclusions

OODBs represent the next step in database evolution, supporting Object-Oriented analysis, design, and programming. They give much better machine performance than RDBs when working with highly complex data structures. However, OODBs will coexist with RDBs because in many situations data independence is more important and efficient than encapsulation. Also, many corporations are locked into RDB systems and would not be cost efficient to switch to an OODB. Military systems have likewise grown over time. Current Theater Air Operations Centers for large fixed theaters receive information from many sources in the form of text messages, image maps, and voice reports. The Oracle RDBMS used by the plans and operations personnel and the Sybase RDBMS used by the intelligence personnel have been maximized to handle a limited subset of the operational requirements. These RDBMS vendors are upgrading their commercial products to provide an Object-Oriented capability to meet the military operational needs.

In the future, OODBs will open up many new horizons and create new capabilities on the Information Superhighway.

References

Abnous, R., and Khoshafian, S. (1990) "Object Orientation - Concepts, Languages, Databases, User Interfaces." John Wiley and Sons, Inc.

Bowman, C. "Database Programing and Design." *Why We Need Object - Oriented Systems* , Feb. 1994, Pp. 27-30.

Codd, E. (1990) "The Relational Model for Database Management: Version 2." Addison - Wesley Publishing Comp.

Jardine, D. (1977) "The ANSI/SPARC DBMS Model." North-Holland Publishing Comp.

Kroenke, D. (1978) "Database - A Professional's Primer." Science Research Associates. Inc.

Landwelar, C. (1988) "Database Security: Status and Prospects." Elsevier Science Publishers B.V.

Learning Tree International, (1991) *Relational Databases: Designs, Tools, and Techniques*. Learning Tree International, Los Angeles.

Lochovsky, F., and Tsichritzis, D. (1977) "Database Management Systems." Academic Press.

Martin, J. (1993) Principles of Object - Oriented Analysis and Design." PTR Prentice Hall.

C PROGRAMMING FOR DIGITAL ANALYSIS,
AND THE UNIX OPERATING SYSTEM

Jonathan C. Bakert

Sauquoit Valley High School
Sauquoit, NY 13456

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1994

C PROGRAMMING FOR DIGITAL ANALYSIS, AND THE UNIX OPERATING SYSTEM

Jonathan C. Bakert
Sauquoit High School

Abstract

In much more depth, the C programming language was studied and was implemented to compose various tools for the ATTI (Analog Test Tool Integration) package under Xwindows. Tools such as a sliding FFT algorithm with a movable 'window' were written along with the C source code for a signal to noise calculator. Also, code to read and write the header file of the Tektronix DAS 9200 (Digital Analysis System) was used to store critical information during analysis. Along with these functions, smaller, more versatile functions were written as part of a larger library to make repetitive tasks, such as the copying of matrices or the summation and squaring of an array, easier. The Unix operating system was also studied and used with more proficiency.

C PROGRAMMING FOR DIGITAL ANALYSIS, AND THE UNIX OPERATING SYSTEM

Jonathan C. Bakert

Introduction

The C programming language has always been an incredibly useful tool for many types of scientific research. In this case, the language was implemented to write various pieces of code for a digital analysis project. Several routines were written for the ATTI (Analog Test Tool Integration) package, such as code to extract header information from the DAS 9200 (Digital Analysis System) for further analysis by other external functions. Also, code was developed to produce data using a 'sliding window' FFT. The portion of data read in by this function would gradually be changed by a user definable setting until the maximum amount of 'windows' had been read in. An FFT would then be preformed and the results would be graphed by a graphing function internal to the ATTI. Signal to noise analysis was also preformed by determining the sum of the squares of the entire signal and then placing it over the sum of the squares of the noise generated.

Furthermore, functions were written to make repetitive tasks easier. A function to calculate the sum of the squares of an array of double integers was written along with a useful function to copy the contents of an array to a destination array. Lastly, the UNIX operating system was studied and the result was a greater proficiency with UNIX commands.

Methodology

When first writing the code to extract the header file from the DAS, the file format, in terms of C, had to be identified. Since code on disk could not be found with all the relevant information on it, the DAS header had to be copied verbatim from a user manual which had a detailed description of its layout and format. Also, the user's manual contained information on the offset between the data structures, which was mistakenly overlooked at first. With this offset and the users manual diagrams, the data structures could be successfully read into memory. With this information in memory, the user and the programmer can utilize information from the data file such as the date it was created, its file type, and the amount of headers which are contained in the file. Also, the frequency can be gathered from analysis of the header file.

However, if the frequency determined from the header file is ever zero, this represents an external signal source, and thus the user of the ATTI package must input it. Along with reading in the appropriate DAS header, our own header had to be supplied to contain data which we needed to access independently. This made our final header file a combination of both our own data and the original DAS generated header.

Below is an example of the code used to read in the information of the DAS header:

```
----

/* readheader.c          */
/* By Jonathan Bakert    */
/* as Part of the ATTI   */
/* Rome Labs/ERDA       */
/* July 20, 1994        */
/* Version 1.1          */

/* Modified Header Format:
 *
 * MEM_HDR
 * (Offset of N bytes)
 * DATA_SET_HDR
 * (Offset of N bytes)
 * DATA_SET_HDR
 * (Offset of N bytes)
 * AUX_DATA
 * USER_HEADER (Defined by ATTI)
 */

#include <stdio.h>
#include <stdlib.h>
#include "fileform3.h"
#include "atti.h"

FILE *Headerfile, *fopen();

int readheader(char *Headfilename, user_struct_p *user_header, mem_hdr_p *memory_hdr, data_set_hdr_p
*data_set, aux_data_p *auxiliary_data)
{
    if((Headerfile = fopen(Headfilename,"r")) !=NULL) {
        if ((*user_header) == NULL) { /* then malloc all structs */
            (*memory_hdr) = malloc(sizeof(struct mem_hdr));
            (*data_set) = malloc(sizeof(struct data_set_hdr));
            (*auxiliary_data) = malloc(sizeof(struct aux_data));
            (*user_header) = (user_struct_p) malloc(sizeof(struct user_struct));
        }

        fread((*memory_hdr), sizeof(struct mem_hdr), 1, Headerfile);

        fseek(Headerfile, (*memory_hdr)->mh_set_hdr.off, 0);
        fread((*data_set), sizeof(struct data_set_hdr), 1, Headerfile);

        fseek(Headerfile, (*memory_hdr)->mh_aux_data.off, 0);
        fread((*auxiliary_data), sizeof(struct aux_data), 1, Headerfile);
    }
}
```



```

    fread((*user_header), sizeof(struct user_struct), 1, Headerfile);

    if (fclose(Headerfile) == EOF)
        return(0); /* error closing file */
    else
        return(1); /* closing is ok */
} else {
    return(0);
}
}
}
---

```

With the `readheader()` function, the memory for the structures was originally allocated while in the function. Since the ATTI handled the memory allocation job, this function had to be passed pointers to the available memory locations.

In C, various functions help in the manipulation and the random access of data. In this instance, the `fseek()` function was used to 'cursor' through the file stream and thus move 'offset' distance from the beginning of the file. This compensated for the offset between each DAS data structure.

For example, though the diagram in the users manual showed this format for the header:

```

HEADER 1
HEADER 2
HEADER 3
DATA
.
.
.
[[END OF FILE]

```

... a more correct representation would be:

```

HEADER 1
(Offset of N bytes between next header)
HEADER 2
(Offset of X bytes between next header)
HEADER 3
(Offset of Y bytes between next header)
DATA
.
.
.
[END OF FILE]

```

Along the same lines was a function to write out the header information which is almost exactly like the

READHEADER.C function except that it implements `fwrite()` instead of `fread()`. Once an `fseek()` was performed in the WRITEHEADER.C function, an `fwrite()` would be executed. `Fwrite()` turned out to be the easiest function in this case, as it could read in the entire contents of a structure with only a single command. The only thing necessary for the programmer was to input a pointer to the data, the number of bytes you wanted to read in, the number of times you wanted to read those bytes, and the file stream.

After the code to write the header had been completed, a 'sliding window' FFT algorithm was worked on. This code read in a certain number of data points which made up the 'window'. The window would then be moved a certain number of bytes over and the data would be read in once again, hence the term 'sliding'. All data acquired this way would be averaged into a single array which would then be multiplied by a 'window' (not to be confused with the sliding window) to eliminate errors when testing. An FFT would then be run on this data and the output would generally be more precise since some of the noise was averaged out with the window passes.

An example of the Sliding FFT code is:

```
/*
  By Jonathan Bakert   08 Aug 94
    for use in the ATTI project

  Returns 0 on error.
  Set tabstop=3
*/

#include <stdio.h>
#include <math.h>
#include <complex.h>
#include <matrix.h>
#include "fileform3.h"
#include "atti.h"

extern user_struct_p user_header;
extern mem_hdr_p memory_hdr;
extern data_set_hdr_p data_set;
extern aux_data_p auxiliary_data;

FILE *fout, *fopen(); /* input file pointer */

int sfft(atti_file_p instruct, atti_gdata_p outdata, int slide_dist, int window_size)
{
    int i, current_pos = 0, times_looped = 0;
    unsigned int num_points;
    double data, mag, win, maxmag, multiplier;
    double noise[1];
    double *time, *temp_freq, *freq, *magfreq;

    fprintf(stderr, "In SFFT now....\n");
```

```

if ((instruct->len==0) || ((fout = fopen ("analysis_data", "w")) == NULL))
{
return(0);
}

num_points = instruct->len;

/* IF PERIOD IS 0, USE USER DEFINED FREQ */
if (data_set->dsh_hdr.dshh_period == 0)
multiplier = ((user_header->Fs)/window_size);
else
multiplier = ((1.0/(data_set->dsh_hdr.dshh_period))/window_size);

fprintf(stderr, "Multiplier = %f\n", multiplier);

outdata->len = window_size/2;
outdata->data = (double *) malloc(sizeof(double) * outdata->len);

for( ; ; )
{
for (i=current_pos; i < window_size + current_pos; i++)
{
data=(double) instruct->data[i];
win = 0 - cos (((double)i / ((double) (window_size - 1))) * 2 * M_PI) + 1;
time[i] = win * data;
}
}

realfft(time, window_size, temp_freq);
MADD(freq, freq, temp_freq, window_size);

times_looped++;
if(current_pos + slide_dist + window_size > num_points)
break;
else
{
current_pos += slide_dist;
}
}

[Portions of Code Removed]

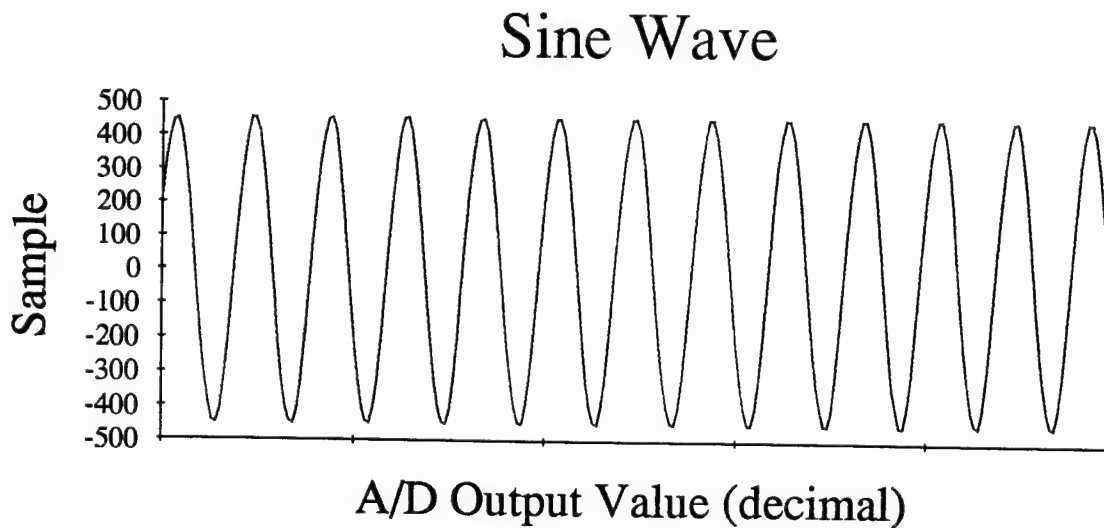
```

A problem with the calculation of the slide distance occurred when the current position was only being set equal to the slide distance. This would create an incorrect next position, and, even worse, would stick the function into an infinite loop. With the quick addition of a plus (+) sign before the equal, this problem was solved.

Also, in the SFFT code, the matrix library of functions was used. These functions were created by the Lab before my apprenticeship. These proved useful in a few occasions. For example, the function MADD would add the contents of array A and array B and place it in array C. The parameter which you needed to pass also included the number of elements which were in the source array. MDIV worked along the same lines as MADD, except that it divided array B by array C,

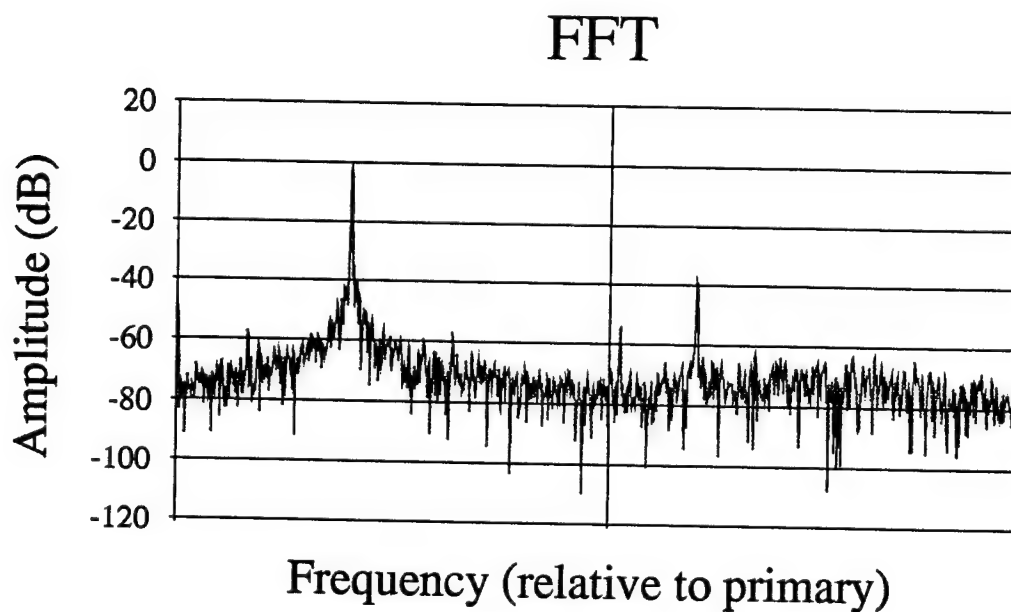
and placed the value in array A. The number of elements in the array also needed to be passed. These elements were represented by the variable S.

The SFFT code would take data such as the sine wave below:



... and then the average of the consecutive FFT's. Also, you'll notice that the sine wave above ends at an irregular point. To correct for this problem, each point had to be multiplied by a set 'window'. This would decrease the amount of noise calculated in the SFFT dramatically.

An example of the data an FFT would send out is:



Next, a signal to noise algorithm was made. This function would be given N number of points. The sum of the squares of these points would then be found and placed over the sum of the squares of the noise generated (the noise being all data not X away from the real frequency.) As a precaution, this function took the frequency as a parameter, but is smart enough to find the real frequency should there be some error, or if it turns out that the user defined frequency is not the most frequent. This function also supports a parameter which would tell the function how many data points to erase on either side of the calculated frequency. For example, if relatively few points were used, the user would want a small number of points erased. If there were a large amount of points, the user would want to 0 out a larger number of points. What was left would be mostly noise.

Lastly, smaller functions were written to copy the contents of an array of doubles to another, and to find the sum of the squares of an array. The sum of the squares of an array was very useful in the signal to noise function which implemented it a few times. This made code look more professional without being cluttered by repetitive statements.

An example of the sum of the squares function is:

```
/* SSQR (Sum of Squares) */
/* By Jonathan Bakert      */
/* Usage:                  */

/* SSQR(Data Array of Doubles, Number of Elements in Array); */
/* Returns the Sum of the Squares (A Double)                  */

#include <math.h>
#include <stdlib.h>
#include <stdio.h>

double SSQR(double *data, unsigned int num_elements);

double SSQR(double *data, unsigned int num_elements)
{
    unsigned int counter;
    double sum_of_squares = 0;

    for(counter = 0; counter < num_elements; counter++)
    {
        sum_of_squares += data[counter] * data[counter];
    }
}
```

```

        return(sum_of_squares);
    }

...

An example of the matrix copy function is:

/* MCPY by Jonathan Bakert */
/* Copies all elements from B array into A array */

/* Usage:
 *
 * MCPY(dest array, source array, # of elements in source
 *
 */
void MCPY(double *a, double *b, unsigned int s);

void MCPY(double *a, double *b, unsigned int s)
{
    unsigned int counter;
    double *a_ptr=a,
           *b_ptr=b;

    int element=0;

    while ( element++ < s)
        *a_ptr++ = *b_ptr++;
}

...

```

Though both of these functions are very simple, they turned out to be very useful in a few different pieces of code. They may also be added to a current library of functions whose purpose is centered around the manipulation of arrays.

In the course of writing these functions, a lot of very useful programming techniques were learned such as passing a variable by 'reference'. When a variable is passed by reference, its address is passed. This allows other external functions to modify data which normally wouldn't be available. This also enforces the idea of modular programming. By having each function do a specific thing, and thereby reducing global data, a new style can be developed that is much easier to understand and much more powerful than languages which tend to produce confusing, entangled code, such as BASIC.

The use of type casting was also learned. This allows the program to successfully modify a variable. For example, an integer and a double integer may be divided by typecasting the integer to a double.

The usefulness of the `malloc()` function and the `calloc()` function was also learned. Since C doesn't necessarily allocate memory space for all its variables or arrays, these functions have to be called to supply a suitable memory space to place the newly acquired values. If these functions are not used prior to declaring a variable or array, it

can have tragic results, such as overwriting highly volatile places in memory. It can also be equally disastrous if you fail to allocate the right amount of memory.

Also, the technique of using the C typedef 'reserved word' when declaring structures also proved very useful. Instead of declaring a structure like this:

```
struct whatever
{
    int whatever[20];
    double hotdog[10];
    etc...
} *rule_world;
```

... it can be shown as less confusing if a structure is declared like this:

```
typedef struct whatever
{
    int whatever[20];
    double hotdog[10];
    etc...
} *rule_world_p;
```

... this way, the word 'whatever' may be used to make code clearer to understand. An instance of it still must be declared by writing something like:

```
rule_world_p rule_world;
```

Now, the structure may be index by using:

```
rule_world->hotdog[]
```

All of these programming techniques were extremely valuable when writing function for the ATTI project over the summer. It's also true that cutting corners in certain areas may get you into trouble! As the old saying goes (or at least sort of goes), for every complex problem there is an easy solution which is quick, neat, and wrong!

Results

When all the work was completed for the ATTI and the program had been sufficiently debugged and tested, a demo was given to a group of people from both Rome Lab, and other locations. To prepare for the presentation, a list of viewgraphs was made with bulleted displays outlining the ATTI and many of its characteristics. Things such as the history of the ATTI and the ATTI's menu structure were discussed, along with an explanation on how it retrieves and manipulates the data. Also, the future uses for the ATTI package were discussed. A few commercial companies are interested in our

package.

Conclusion

Over the course of the summer, much was learned about C programming and its uses. New ways of writing code and using functions were found, along with a helpful insight on how work in the technological field will be. Also, a few examples were shown on how algorithms can be implemented in programs to make work both easier to understand, and more effective. It was also apparent that writing your own functions, in the long run, can save you a lot of time when you try a larger project later on. By writing small functions to take care of repetitive tasks, the size of code can be cut down dramatically. The code will also be more appealing to look at, since things won't be as cluttered with one line function calls.

The nature of data acquisition and analysis was also shown to me, and its methods described. It was also enjoyable to work in a team to get a project assembled. After every year that passes, I learn more and more about the C programming language and different ways to use it. Since I plan on entering a software field, any type of language (especially C) will be very beneficial to getting a job later on.

Also learned were a lot of things about the UNIX operating system. A whole new series of commands was found that turned out to be very useful in transferring files to and from different directories. Getting useful software libraries was also possible with the help of internet access. And as a bonus, having previous knowledge of the OpenWindows and UNIX operating systems will give me a head start in college from those who are new to UNIX's style of commands. Hopefully, I can continue learning the C language and eventually get some formal instruction so that I may learn to write larger, more complex programs. Since software engineering rarely relies on one person to develop something, having prior experience will make it much more easier to adapt in a job setting where many people contribute to a specific goal.

References

1. Peter Aitken & Bradley Jones. Teach Yourself C in 21 Days. SAMS Publishing. A Division of Prentice Hall Computer Publishing, 1992.
2. SunOS User's Guide. Sun Microsystems, Inc., 1989.
3. SunOS: An Introduction. Sun Educational Services Revision A, 1989.
4. SunOS: Doing More. Sun Microsystems, Inc., 1989.

**A STUDY IN THE DEVELOPMENT OF SPECIALIZED SOFTWARE
FOR PRI AND HISTOGRAM GENERATION IN SUPPORT OF
SIGINT RESEARCH AND DEVELOPMENT EFFORTS.**

Craig M. Belusar

**Oneida High School
501 Seneca St.
Oneida, NY 13421**

**Final Report for:
High School Apprentice Program
Rome Laboratory**



**Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC**

August 1994

A STUDY IN THE DEVELOPMENT OF SPECIALIZED SOFTWARE FOR PRI AND HISTOGRAM GENERATION IN SUPPORT OF SIGINT RESEARCH AND DEVELOPMENT EFFORTS.

**Craig M. Belusar
Oneida High School**

Abstract

The goal of this project was to produce software that would generate random generated Pulse Repetition Intervals (PRI's) and display a graphic representation of the data. The solutions to this problem included learning about computer hardware and software, radar signal propagation, and computer graphics. The result of this project was a set of programs that generated random PRI's based on specified parameters with the ability to display the PRI's in a histogram and / or pie chart display. Solving these problems and writing the software increased my knowledge of computers and software development to a great extent. It also proved beneficial to the ELINT Development Facility (EDF) at Rome Laboratory as it aided in their research in signal intelligence.

Table of Contents

Introduction	3 - 4
Problem Description	3 - 4
Methodology	3 - 6
Results	3 - 8
Conclusion	3 - 9
Appendices	
A. PASCAL program for PRI generation	3 - 11

A STUDY IN THE DEVELOPMENT OF SPECIALIZED SOFTWARE FOR PRI AND HISTOGRAM GENERATION IN SUPPORT OF SIGINT RESEARCH AND DEVELOPMENT EFFORTS.

Craig M. Belusar

Introduction

The number of applications for computer software and hardware is unlimited. Included among these applications are programs that aid signal and electronic intelligence research efforts. The study of radar waves and their characteristics rely greatly on signal processing techniques. Much of this processing is now done with specialized algorithms and software.

Problem Discussion

The ELINT Development Facility (EDF) is currently developing algorithms that analyze the Pulse Repetition Interval (PRI) of radar waves. For test purposes, it would be desirable to have sets of PRI values with known characteristics to experiment with. The problem I encountered was to develop software that would provide these PRI values. The

software consists of two programs; the first program was made to generate random PRI's, and the second is used to display the data in a graphical manner as a histogram.

Generating random PRI's requires a basic understanding of radar wave forms. In a radar wave form, a PRI is the duration of time from the start of one pulse to the start of the next. It is the distance on *figure 1* from point A to point B represented mathematically as:

$$f_x = \text{radar crystal frequency};$$

$$N = \text{countdown};$$

$$PRF = f_x / N$$

$$PRI = N / f_x = 1 / PRF$$

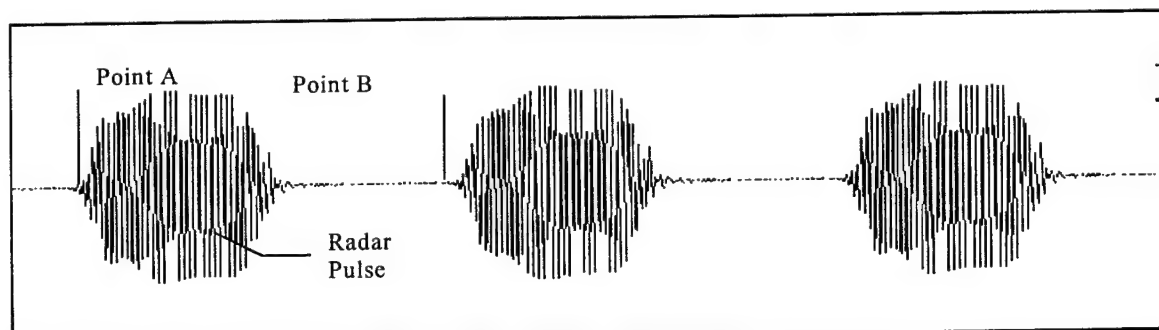


Figure 1. Representation of a PRI in a Pulsed Radar Waveform.

A crystal is the component of a radar transmitter that controls the rate of pulse transmission, known as the Pulse Repetition Frequency or PRF. The PRF, measured in pulses per second (pps), is equal to the crystal frequency divided by the countdown. The

PRI is the inverse of the PRF. As such, the PRF is inversely proportional to the countdown, N, and the PRI is proportional to N.

The program that was developed to solve this problem used both a specified frequency and variance to produce a designated number of possible PRI's. The variance is the amount in which the actual PRI can vary from the theoretical PRI. In the real world, this variance can be caused by circumstances such as environmental factors or aging equipment.

Software was also developed to take the PRI values generated previously and display them as a histogram or pie chart. A histogram compares the number of bins to the amount of items in each bin. The histogram algorithm takes the list of PRI's and assigns each one a bin number. The bin numbers are represented by the x - axis, and the y - axis is represented by the number of values in each bin. The bin size was calculated by subtracting the upper value of the range by the lower and dividing by the number of bins. This is the width of the bin, or delta. The bin number was then calculated by subtracting the lower limit by the value and dividing by delta. The number of values in each bin are calculated by counting the PRI's that fall within the range (delta) of a particular bin.

$$\text{delta} = (\text{upper} - \text{lower}) / \text{number of bins} ;$$

$$\text{bin number} = \text{value} - \text{lower} / \text{delta};$$

Methodology

Two programs had to be written to solve this problem. The first was written in the programming language Pascal, the second was written in the C programming language. To accomplish this, I needed to learn how to program in these languages as well as develop algorithms to solve the problems. Both Pascal and C are block structured programming languages that include independent structures, called functions in C and procedures in Pascal, which link together to form a whole. By dividing the program into these functional units, it becomes easier to develop and maintain the software over the life-cycle of a project.

The PASCAL program which calculated the PRI's involved concepts that I had previously been unaware of including radar theory, file input and output, random number generation, and using specific functions in Pascal that were essential to the final program. It also required the use of many programming language constructs such as loops, data types, and arrays. This expanded my programming knowledge and prepared me for the challenge of learning the C language and developing software in C.

The second program was used to generate a histogram display (see figure 2) of the PRI's created with the PASCAL program, and was much larger and more complicated than the previous program.

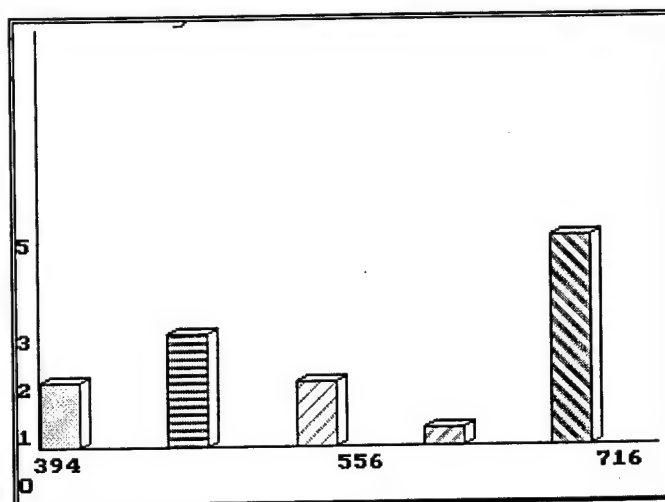


Figure 2. Example Output from Histogram Program.

This program read input from a file that contained sequences of numbers (PRI's), and produced a histogram as well as a pie chart to display the information. The histogram displayed the number of bins and the range on the x - axis, and the number of values in each bin on the y - axis. It also took information

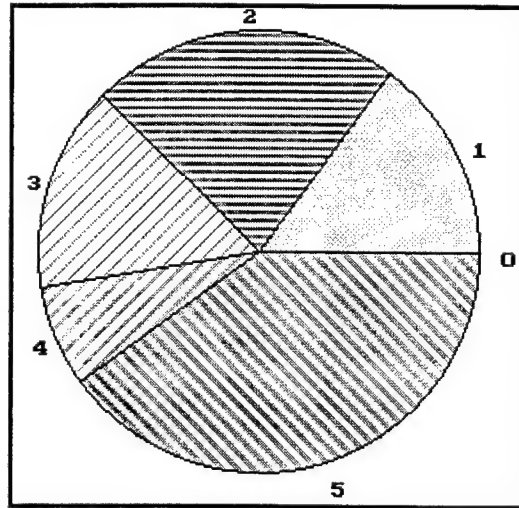


Figure 3. Pie Chart from Histogram Program.

such as the file that the user wanted to read from, whether or not the user wanted to enter the data range manually or have it calculated automatically, and also whether or not the user wanted the pie chart (see figure 3) and the bar graph to be displayed on the same or separate screens.

Writing this program involved a lot of research as a result of having to learn the C programming language. It also involved many new concepts including pointers, graphics, structures, as well as histograms. Each of these concepts proved useful as I learned to join them to form a well-organized and maintainable program.

Results

The project produced two valuable computer programs for use in the EDF's research on radar PRI analysis, and helped me to learn valuable programming skills that are useful in college as well as the work place.

For example, one major part of programming as equally important as the results of the program is the organization and design of the software. This is where I have

improved the most. By making my program more modularized, so the functions act independently of each other, I have greatly improved my programming efficiency, organizational abilities, and problem solving skills.

The EDF has benefited from this project as it provides them with a method for easily creating a set of PRI's that are useful for testing and experimentation. The data has already been used in other EDF research efforts that are developing techniques for determining the crystal countdown of a radar from PRI information.

Conclusion

The project assigned to me was a success. It not only produced useful results for the EDF / Rome Labs, but also improved my programming abilities and general problem-solving skills. It is something that I am sure to use in college as well as my career as an engineer. The project also increased my knowledge of computers in general as I learned to install the necessary software on my assigned computer and configure the hardware. Another important and beneficial result of this experience was the insight I gained about working in a true engineering environment as an individual and part of a group.

APPENDIX A

PRI generation program written using Turbo PASCAL

```

(*****)
(*)
(*)
(*)          PRI Generator
(*)
(*)          The purpose of this program is to generate random PRI's for
(*)          SIGINT research and development
(*)
(*)
(*)          Craig Belusar
(*)          July 10, 1994
(*)
(*****)

```

Program Ra(Input,output);

Uses

Crt;

Var

f:text;

amount:integer;

t : real;

out:string;

list:array[1..100] of real;

variance:real;

frequency:real;

(*****)

Procedure EnterInput(var amount:integer;var out:string;var variance,frequency:real);

Begin

Writeln ('Please enter the desired amount of PRIs.');

Readln (amount);

Writeln ('Which file would you like to send the output to?');

Writeln ('Please include path, and complete file name.');

Readln(out);

Writeln('Please enter the desired variance.');

Readln(variance);

Writeln('Please enter the desired crystal frequency.');

Readln(frequency);

End;

(*****)

Procedure random_numbers(var f:text; var frequency:real;var variance:real;amount:integer);

var

count:integer;

number:real;

Pri:real;

multiple:integer;

Begin

Arc-Second Raster Chart/Map Digitized Raster Graphics
Data Exploitation

Shawn H. Bisgrove

Rome Free Academy
500 Turin Street
Rome, NY 13440

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

August 1994

Arc-Second Raster Chart/Map Digitized Raster Graphics Data Exploitation

Shawn H. Bisgrove
Rome Free Academy

Abstract

A program was developed using C in a UNIX environment on a SPARC 10 running SunOS 4.1.3 to efficiently exploit ARC Digitized Raster Graphics from a mounted compact disc and create multiple outputs. Utilizing SyBase's Structured Query Language as a standardized storage method this program populates the database tables. A raw output and a formatted output can also be created to assist in cataloging ADRG material. This program is useful when attempting to catalog large collections of ADRG cartographic material.

Arc-Second Raster Chart/Map Digitized Raster Graphics Data Exploitation

Shawn H. Bisgrove

Introduction

The Air Force Geographic Information Handling System (AFGIHS), created to access digitized cartographic versions of physical map data obtained by the Defense Mapping Agency (DMA), lacked the ability to quickly remove pertinent data from a compact disc using the ARC Digitized Raster Graphics (ADRG) format and display the data footprint in the Spatial Display Tool (SDT).

According to document PS/2DJ/100, the first revision of ADRG specifications dated April 1989, and the superseding release Mil-A-89007 dated February 1990 as released by DMA state, "The equal Arc-Second Raster Chart/Map provides for a rectangular coordinate system and projection system at any scale for the entire ellipsoid based on the World Geodetic System 1984. An ARC Digitized Raster Graphics consists of Digitized Raster Graphics transformed into the ARC system and accompanied by ASCII encoded support files." The fourth edition release of Digitizing the Future, a publication by DMA, estimates around that 4000 ADRG titles are available representing 14,000 scanned paper maps. ADRG information is stored on a ISO-9660 compact disc using ISO-8211 ASCII data. These standard specifications insure the wide use of ADRG compact discs in almost any system produced at this time.

As of release 3.0 beta, only two programs, the Image Index Support SyBase Compilation (Iis), not to be confused with Iis flatfile compilation, and the Map Administration Tool (MAT), used with the Mapping Applications-Client/Server (MACS), relied on SyBase as a standardized link for manipulation, storage, and retrieval of information. These MACS programs are designed for a multitude of map display and manipulation techniques and, therefore, these programs require a graphical interface for the displaying of digitized maps. SyBase, an interactive client server, developed by SyBase Incorporated, uses a database type entry of data at both a programming level and at a user interactive level. This combination allows for the creation of a host of programs to work with the interactive user environment. This entry of data can be queried against or recalled at any time and is uniform so database structures can be changed. SyBase, since it is not a graphic dependent program, opens up the possibility of populating these database tables with text only system with a CD-ROM drive and recalling the information on a networked machine capable of a Graphical User Interface (GUI) required by AFGIHS.

Discussion of Problem

With older programs, problems occurred when ADRG compact discs contained multiple distribution rectangles or multiple zone distribution rectangles. These programs used fixed hard coded design and did not account for the non-standard contents of ADRG such as JNC's that contained polar charts and JOG's with incompatible data. The coding of CDREAD allows the program to work in any given situation with an ADRG standard compact disc as specified by the ADRG specifications. The compact disc follows a hierarchical file design which creates logical breaks in the specific information. The header file is of variable length which means it changes in size and content depending on the number of distribution rectangles that exist. This program is designed to read the file not as a fixed layout but rather a variable length. This design allows for a near logical header extraction of data from the individual files.

With IIS data footprints of an ADRG could be manually entered and displayed. This process wasted both time and energy and required the user to have knowledge of cartographic map data. The user would need to know the proper placement of the coordinates. Once enter, however, the user could view the area encompassed by the ADRG.

In some cases a distribution rectangle will have multiple compact disc directory entries. The specification explains for these distribution rectangles have multiple source graphics. Multiple source graphics would occur in cases where a physical map was too big for physical scanning into one file, sheets with multiple datum's, and sheets with special processing requirements. These would all create special situations when obtaining information for the compact disc.

Methodology

Using SyBase's C language developing libraries and include files as well as the standard UNIX include files and libraries, an application called CDREAD was created to quickly modify existing SyBases tables as well as creating a more extensive set of table and subsequently update with information from other ADRG compact discs as it is added. This application was designed so that datafields found in the files of an ADRG are subsequently cataloged inside appropriate SyBase tables. This creates a standardized link that current and future MACS programs could use to create a complete footprint in the SDT including many aspects unique to the ADRG format such as product type. The program that has been created updates the Image IIS tables so that a basic red footprint can display the covered area contained on the ADRG compact disc. The ADRG information is stored in the IIS tables until the user deletes the specific information and/or truncates the tables. This information then can be queried and manipulated using the SyBase's

Interactive Structured Query Language (ISQL), IIS SyBase compiled, or with another third party program designed to work with the specific SyBase tables.

The code was developed using linked lists in order for the dynamic "on the fly allocation" of memory to occur. A linked list is created to contain the distribution rectangle information and contains two pointers to the linked list of charts. One pointer acts as a current placement in the list and the other as a head or root placement. This allows the program to efficiently access data within a linked list containing the information after population occurred. Using pointers, management of the linked lists becomes very simplistic and can quickly and logically allocate just the memory needed in any given scenario. These linked are **malloc**'ed as information was obtained so that memory would only be used if data existed for the fields. With the use of pointers distribution rectangles are read in and **malloc**'ed first, then once completed the charts are allocated and read in. This follows the hierarchical structure that simplifies ADRG data exploitation.

Memory problems would be a very rare occurrence due to CDREAD coding that allow an allocation of memory on demand. This "on the fly allocation" will accommodate for any ADRG compact discs no matter how many distribution rectangles or charts exist. Also a standard or error exit does not leave inaccessible links to memory blocks thus creating a memory problem when the program is executed multiple times. The limitation with memory allocation is that when no more is allocable the program will compromise some but not all the data. This would only be a problem on systems which are taxed with a full memory load.

The code itself is hierarchically design to match the file hierarchical design contained on the ADRG compact disc. This hierarchical file format, which is required to load information from an ADRG and gives the program maximum results and without using excess memory. The compact disc's header file known as TRANSH01.THF contains information necessary for finding and obtaining the distribution rectangle and subsequently the chart files. These distribution rectangles follow the ADRG directory and file name convention. The convention **ssccdd** where **ss** is the chart series code, **cc** is the country code, and **dd** is the distribution rectangle number is used in all ADRG compact discs. This naming convention is used when reading in files specific to a particular distribution rectangle and chart.

All of these files, including the TRANSH01.THF, do not have a preset limit, or a specific order, so in programming a code needed to be developed that would be flexible. When CDREAD opens files containing information, the ADRG naming convention is used. This naming convention **ss** is the chart series code, **cc** is the country code of the northwestern most source graphic, and **dd** is the distribution rectangle on each CD-ROM. For each distribution rectangle four files and a directory exist. These files

contain specific information used in identifying the current distribution rectangle or chart. The file `ssccdd01.GEN` contains all the general information pertaining to the distribution rectangle. The `ssccdd01.QAL` file contains a security and release field, a up to dateness field and a horizontal and vertical accuracy field. The `ssccdd01.OVR` stores the overview image information. The `ssccddzz.IMG`, where `zz` corresponds to the ARC Zone Number in which the image data exists, stores the scanned graphic. The directory `ssccddgg`, where `gg` corresponds to the number of the northwestern most chart, starting at one and increasing sequentially, contains all chart information. In this directory is the `ssccddgg.SOU` file which contains information regarding individual charts. Some ADRG compact discs have different countries on the same disc so it was not reliable to hardcode in the `sscc` obtained from the first distribution rectangle. This field is exploited from the header file and placed in the distribution rectangle fields first. A compact disc could for example contain four distribution rectangles `JABD01`, `JAAZ02`, `JAAZ03`, `JAAZ04`. This would create a problem if the program attempted to assume `JABD` for all four distribution rectangles.

Once all pertinent ASCII data has been read from the compact disc, `CDREAD` proceeds to dump information to the files selected by the user at the time of execution. Two forms of ASCII dump files are created of the ADRG compact disc. A raw dump of all information occurs as soon as the information populates the memory structure and a file is written, and subsequently updated with fields, until all data is received. The layout of the raw ASCII dump is documented in appendix A. At this point the stream is closed and the file is written using the standard UNIX commands `fprintf` and ultimately the stream is closed with `fclose`. The raw dump contains no headings for the values but is useful when information is not contained in either the formatted or SyBase dump. The raw dump contains all information about the compact disc, distribution rectangles and charts. A formatted ASCII dump is created to be ultimately printed out or evaluated to decide if to load the information from the compact disc into SyBase tables. The formatted ASCII dump contains some of the fields as outlined in appendix A. This information could be conceivably used to create a third party utility to populate SyBase tables with specific information. With these two dump files, the data is reset before each compact disc is exploited. This insures the files will not become too large for any particular system. If this information cannot be written to the `cdread/runtime/dump_files/` directory then the information is defaulted to the current directory. This could occur due to many reasons. Two of these reasons would be that the directory has been deleted and must be recreated or a lack of rights to access the directory. This is a problem that can be resolved by your super-user granting directory read and write access to the appropriate users.

When populating SyBase, `CDREAD` updates six distinct tables as documented in appendix B. `CDREAD` updates three tables used in conjunction with `Iis` and updates three more tables to be used by future programs. With `Iis` the `image`, `image_comments`, and `image_asc_be` tables are manipulated. `Iis` originally used the tables to display a graphical footprint outlined in red based according to the

coordinates, a picture in a popular image format that would be contained in the footprint, the type image, and the directory where the image exists for viewing purposes. With ADRG's however, the tables will be used just for displaying footprints.

The **image** table, used by Iis SyBase compiled, contains information about the distribution rectangle. The **image_comments** table contains the specification identification as well as the corresponding charts series designation, unique source identification, source edition identifier, and significant date. The **image_asc_be** table is used just to help keep track of the number of compact discs owned of the same type in the collection. Three other tables **cdread_main**, **cdread_dr**, **cdread_chart** are manipulated. The **cdread_main** table contains information about the compact disc. The **cdread_dr** table contains one entry per distribution rectangle with the same query as the main information existing in each entry. The **cdread_chart** table contains one entry per chart existing in the distribution. This table has the same query information existing for the compact disc general information as well as the distribution rectangle so that a common query point is maintained when searching for specific information. Once done manipulating tables, the program proceeds to updates the statistics in each of the SyBase tables so that a chance of error when querying the field is non-existent.

In evaluating the data retrieved from the compact disc a few conversions are necessary. CDREAD extracts the data from the compact disc in the form of strings which are stored as the string length plus a null terminator. This method is the easiest to read from a file byte by byte. Once a link between CDREAD and SyBase is created many of the entries are converted from strings to integers. The conversion for degree notation is used on all coordinates stored in the SyBase tables. The ADRG format specifies the use of DDMMS.SS for latitude and DDDMMSS.SS for longitude whereas SyBase tables use a converted integer. The formula $[\text{value} = 3600 * (\text{sign} * (\text{deg} + \text{min} / 60 + \text{sec} / 3600))]$ is used to convert to the integral storage point and a conversion can be made back to DDDMMSS for longitude and DDMMS.SS for latitude. In order to make this conversion the program should divide by 3600, obtaining a double (ex. 10.508333). Next a parsing of the integral value will result in the degrees (ex. 10). The decimal portion of the number then is multiplied by 60 (ex. $0.508333 * 60$). The next integral value is the minutes (ex. 30). Once the integer is removed the remaining decimal is multiplied by 60 (ex. $0.4999 * 60$). This last integer is the second of the DMS coordinate. The program also needs to account for the sign and assign the proper cardinal direction to the coordinates. This method loses the .SS for storage purposes and does not affect the accuracy of the footprint. ADRG compact disc's generally tend to contain zeros in the .SS place depending on the accuracy and size of the map. In the program implementation a basic parsing technique is used to obtain the degrees, minutes, and seconds.

A problem inherent in the programming design would be the lack of duplicate error checking. With CDREAD no comparative query is done before information is exploited and added to the SyBase tables. There is currently no functionality for resolving redundant entries. This poses no problem in the actual search and query of an entry except Iis SyBase Compile will just display the redundant entries.

This program has been designed to work with any ADRG compact disc following the ADRG specifications. This program does correct the pitfalls of earlier programs with fixed hardcoding. With the ARC digital raster graphic information now able to be displayed using Iis or a third party program a person can quickly sort or find an ADRG compact disc in a collection. Using a query within Iis, a user can quickly see if a compact disc for the area exists as well as how many copies. From that point a determination can be made on what is to be done with the data.

Conclusion

With past programs flexibility was a trait to be desired in a software application. This led to complex battles with an operating system to place these programs online. This lack of flexibility was a disadvantage when trying to load in ADRG material. Past programs also only worked with existing software rather than offering a bridge to expand upon. Since CDREAD works with Iis as well as leaving a link to a new program, making the design capable of expansion. Using the formatted ASCII dumps a user can quickly and easily visually retrieve information about the ADRG compact disc. The flexibility of the program as well as the ease of compilation ultimately should create a better organization of ADRG material used.

With a conversion over to digital cartographic maps a need for a cataloging program has been created. This need based on the collection sizes of today's digital resources if compounded by older methods of organization. With ADRG material organization of the compact discs becomes a hassle because of the quantity as well as replacing the out of date material with new material. A future program could be designed to query existing compact discs of a certain age. Once displayed a user could physically remove the compact discs using the ADRG title and series designator displayed and replace them with new more up to date revisions. This would provide a cost effective way of managing collections of ADRG material in existence.

In the future, programs relying on CDREAD hopefully will be used to quickly catalog ADRG compact discs into a SyBase table which a third party program similar to Iis could quickly display selective information. Examples would include displaying all of the JOG-A's in blue and all the TPC's in green. Certain out of date material would, when queried, be displayed in a thatched rectangle so a user can maintain

an extensive collection of ADRG material with little or no effort. This would create a visually assessable list of coverage by a particular type of chart. This has unlimited possibilities and with the included source code modifications can be made when specifications are changed or if need warrant it.

CDREAD due to its simplicity of code and flexibility in setup can be updated and customized to fulfill any front end capacity. This program contains traits which make it an ideal stepping stone to the future of ADRG data exploitation.

References

Digitizing the Future 4th Edition, Defense Mapping Agency Aerospace Center, Missouri. 1993.

Kernighan and Ritchie. The C Programming Language, PRENTICE-HALL, INC., Englewood Cliffs, New Jersey. 1979.

MIL-A-89007 ADRG Specifications Second Edition, Defense Mapping Agency Aerospace Center, Missouri. 22 February 1989.

PS/2DJ/100 ADRG Specifications First Edition, Defense Mapping Agency Aerospace Center, Missouri. April 1989.

**AN IMPLEMENTATION OF THE
MULTIPLE SIGNAL CLASSIFICATION ALGORITHM (MUSIC)
IN MATLAB**

Stacy Fitzsimmons

**Vernon-Verona-Sherrill Central High School
Rte. 31
Verona, NY 13478**

Final Report for:

**High School Apprentice Program
Rome Laboratory (AFMC)**

Sponsored by:

**Air Force Office of Scientific Research
Bolling Air Force Base, DC**

and

Rome Laboratory (AFMC)

August 1994

**AN IMPLEMENTATION OF THE
MULTIPLE SIGNAL CLASSIFICATION ALGORITHM (MUSIC)
IN MATLAB**

Stacy Fitzsimmons
Vernon-Verona-Sherrill Central High School

Abstract

An algorithm to separate multiple signals on an antenna using spatial processing was implemented in MATLAB using concepts from linear algebra. A simulation was created with a given number of antenna array elements and a given number of incoming signal wavefronts, accompanied by noise. Functions were written to generate the signal environment, the noise waveforms, the array manifold, and the array data in order to set up the simulation. The MUSIC (Multiple Signal Classification) algorithm was used to calculate the angles of arrivals of the incoming signals so they could be separated from the other incoming signals and processed.

AN IMPLEMENTATION OF THE MULTIPLE SIGNAL CLASSIFICATION ALGORITHM (MUSIC) IN MATLAB

Stacy Fitzsimmons

Introduction

Collecting signals from an airborne platform presents a number of challenges. Perhaps the most challenging problem stems from the fact that the platform will generally receive a multitude of signals, all coming in from different angles. In order to process any one of these signals, we need to have a way to locate a specific signal and separate it from the other incoming signals. Conventional processing approaches such as time-domain and frequency-domain processing will not work to separate just one signal because many or all of the incoming signals can occupy the same frequency bandwidth. However, if the signals each arrive at the platform with a different *angle of arrival* (AOA)¹, we can use a *spatial* processing approach. To do this, instead of using a single antenna, we use an *antenna array* with a known geometry. With the spatial diversity provided by the antenna array, we can calculate the signal AOAs. Therefore, the signals can be separated from one another and processed. One algorithm which performs this calculation is the MUSIC (Multiple Signal Classification) algorithm.

Discussion of Goals and Objectives

The IRAA COMINT Group at Rome Laboratory is quite interested in spatial processing algorithms, since many of their systems are designed to be used from airborne collection platforms. They wanted a capability to simulate these algorithms in-house in order to gain a better understanding of how they worked. Since they use MATLAB extensively for in-house research, and since MUSIC is such a fundamental algorithm for spatial processing, the overall goal of the apprenticeship became to produce an implementation of the MUSIC algorithm using MATLAB.

This involved first becoming familiar with the operation of MATLAB and its associated toolboxes (in particular, the Signal Processing Toolbox), and the general concepts of linear algebra. Then the fundamentals of the MUSIC algorithm were reviewed, using reference [3] and some help from the technical consultants. Finally, the MUSIC algorithm itself was implemented as a MATLAB function.

¹For this report, we will only be concerned with *azimuthal* angle of arrival; we do not consider the effects of different *elevations* (see Appendix A). If we wanted to consider these effects, the array manifold (discussed later) would become a function of both AOA and elevation.

Before we can explain the MUSIC algorithm, we first need to have a model of the data received by the antenna array. Figure 1 shows a snapshot in time of the incoming signal wavefronts and the receiving antenna array elements (in this example, we are using a linear array). The antenna array contains a variable number of elements, M . The number of these array elements must be greater than the number of incoming signals, K , so that all the signals will be accepted. If M is less than K , the array is considered to be "overloaded," i.e. there would be more unknowns than equations to solve them. The signal wavefronts all have different angles of arrival (AOAs) at the antenna array elements, and are accompanied by a spatial noise field (due to environmental noise, for example). The antenna array samples this signal + noise environment periodically in time.

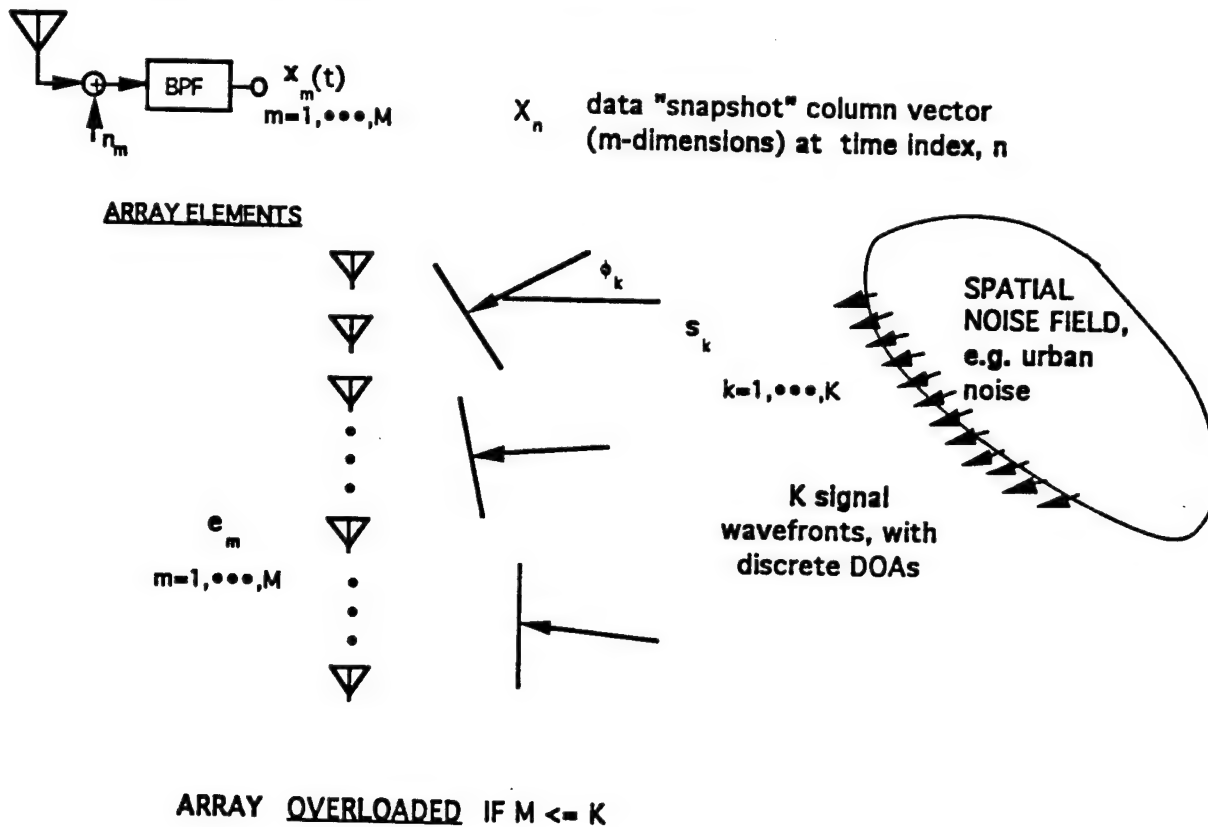


Figure 1: Collection Scenario

In addition to modeling the signal environment, we also need to model the antenna array itself. In order to do this, we must "calibrate" the array — that is, send a known signal at it from a known AOA, and measure the actual outputs of each of the array elements as compared to the original known signal. This will give us array calibration data for a single AOA. Using this process for every possible AOA

between 0° and 360° (using a fixed step size, called the resolution), we build up a table of array calibration data, also known as the array manifold.

Figure 2 shows the array manifold for an array with 3 elements. In this case, the array calibration data for a single AOA is simply a point in 3-dimensional space, since it represents the outputs of each of the 3 antenna elements. Thus, the array manifold becomes a “rope” in 3-space.

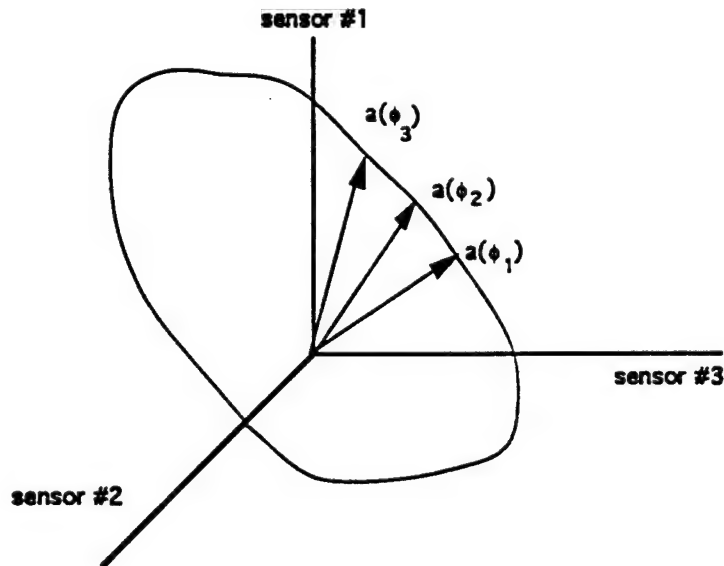


Figure 2: Array Manifold

Finally, the actual output of the array elements can be expressed as a combination of the above components, *i.e.* the incoming signals, the array manifold, and the noise. In particular, each signal is scaled by the array manifold vector corresponding to its AOA (these vectors are called the *steering vectors*). The noise associated with each array element is added to this.

Figure 3 shows the output of an array with 3 elements. Since the data received by each of the array elements at a particular time can be represented as a point in 3-space, the data received over time becomes a curve winding through 3-space.

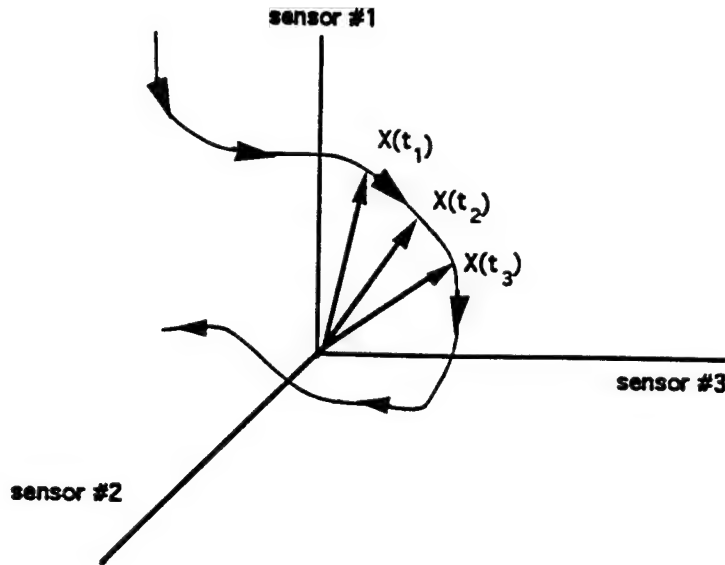


Figure 3: Data Received by Antenna Array

Matrix Representation

The above data model can also be explained by using matrix equations. This is particularly handy, since MATLAB is designed to operate on matrices directly. Let S be the signal matrix. Each column of this matrix is a snapshot in time of the K signals. If we assume that we have N snapshots in time, S is K by N .

$$S = \begin{bmatrix} s_1(t_1) & s_1(t_2) & \dots & s_1(t_N) \\ s_2(t_1) & s_2(t_2) & \dots & s_2(t_N) \\ \vdots & \vdots & \ddots & \vdots \\ s_K(t_1) & s_K(t_2) & \dots & s_K(t_N) \end{bmatrix} \quad (1)$$

The spatial noise field can be modeled by another matrix, V . Since the noise is assumed to be present at each array element, V will be M by N . Noise is the interference that might occur with the signal. We will assume that the noise is white Gaussian noise, and that the noise at each array element is uncorrelated. Each column of V shows a snapshot in time of the noise at each array element.

$$V = \begin{bmatrix} v_1(t_1) & v_1(t_2) & \dots & v_1(t_N) \\ v_2(t_1) & v_2(t_2) & \dots & v_2(t_N) \\ \vdots & \vdots & \ddots & \vdots \\ v_M(t_1) & v_M(t_2) & \dots & v_M(t_N) \end{bmatrix} \quad (2)$$

The array manifold can be expressed as a matrix \mathbf{a} , where each column represents the complex array response for each array element at a particular AOA. Thus the matrix has M rows, but the number of columns is dependent of the resolution of the array calibration. For example, if the resolution of the calibration is 1° , \mathbf{a} will have 360 columns, while if the resolution is 0.5° , \mathbf{a} will have 720 columns.

$$\mathbf{a} = \begin{bmatrix} a_1(\theta_1) & a_1(\theta_2) & \dots & a_1(\theta_{360/\text{resolution}}) \\ a_2(\theta_1) & a_2(\theta_2) & \dots & a_2(\theta_{360/\text{resolution}}) \\ \vdots & \vdots & \ddots & \vdots \\ a_M(\theta_1) & a_M(\theta_2) & \dots & a_M(\theta_{360/\text{resolution}}) \end{bmatrix} \quad (3)$$

We now let \mathbf{A} be the array manifold vectors corresponding to the AOAs of the signals in \mathbf{S} . Thus, \mathbf{A} is M by K . These vectors are also called the *steering vectors*. In this equation, $\phi_1, \phi_2, \dots, \phi_K$ represents the AOAs of the signals in \mathbf{S} .

$$\mathbf{A} = \begin{bmatrix} a_1(\phi_1) & a_1(\phi_2) & \dots & a_1(\phi_K) \\ a_2(\phi_1) & a_2(\phi_2) & \dots & a_2(\phi_K) \\ \vdots & \vdots & \ddots & \vdots \\ a_M(\phi_1) & a_M(\phi_2) & \dots & a_M(\phi_K) \end{bmatrix} \quad (4)$$

Finally we get to the representation of the data actually emitted by the array elements. From the previous description, we know that we have to first scale the incoming signals by their associated steering vectors, and then add the noise associated with the spatial noise field. This data matrix, \mathbf{X} , can be calculated by:

$$\mathbf{X} = \mathbf{A}\mathbf{S} + \mathbf{V} \quad (5)$$

Thus, \mathbf{X} is an M by N matrix whose columns are each a snapshot of the received data at each array element.

$$\mathbf{X} = \begin{bmatrix} x_1(t_1) & x_1(t_2) & \dots & x_1(t_N) \\ x_2(t_1) & x_2(t_2) & \dots & x_2(t_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(t_1) & x_M(t_2) & \dots & x_M(t_N) \end{bmatrix} \quad (6)$$

Now that we have a description of the data model, we can discuss the MUSIC algorithm. MUSIC uses a *subspace* approach to separate the individual signals from each other and the noise. What exactly does that mean? We can think of the M-dimensional data signal given above in (6) in terms of a vector space; in this case, it would be an M-dimensional vector space. Given this M-space, MUSIC tries to separate the signals and noise into two *orthogonal* subspaces — the signal subspace and the noise subspace. The received signals must lie in the signal subspace. Thus, to calculate the signal AOAs, we simply need to find where this signal subspace intersects the array manifold. The vectors to these intersection points on the array manifold are simply the steering vectors. This situation is illustrated in Figure 4 for an antenna array with 3 elements, and two signals. Thus the signal subspace is a plane (2 dimensions, since we have 2 signals) and the noise subspace is a line (3 total dimensions - 2 signal dimensions = 1 dimension) that is orthogonal to the signal subspace. The vectors $\mathbf{a}(\phi_1)$ and $\mathbf{a}(\phi_2)$ are the steering vectors. We will refer back to this figure in the subsequent description of MUSIC.

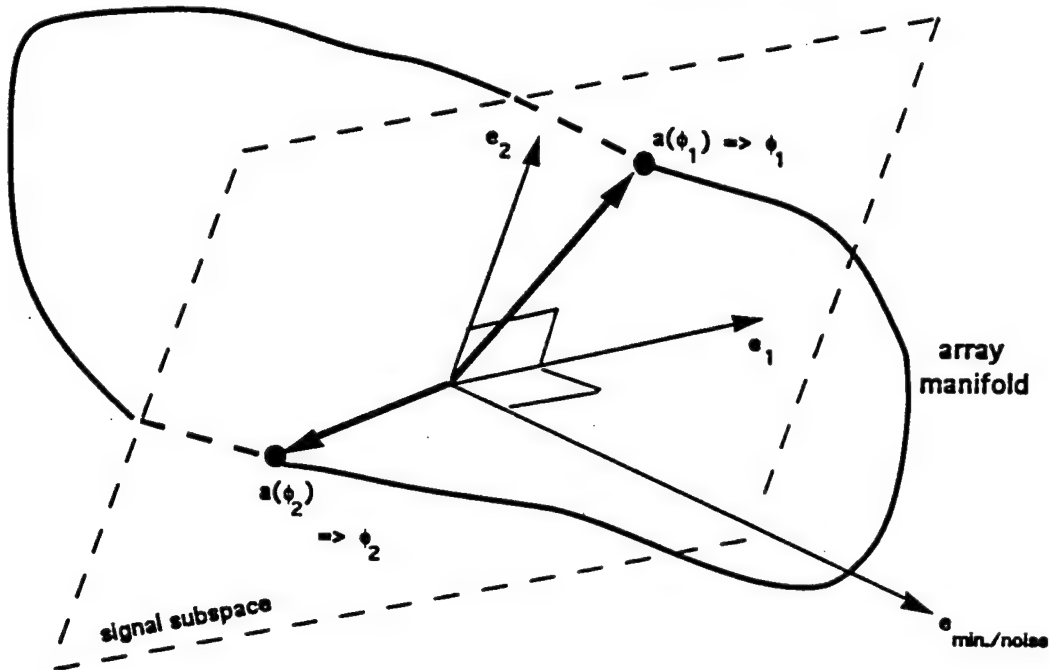


Figure 4: MUSIC Signal and Noise Subspaces

The only information the MUSIC algorithm needs is the array manifold \mathbf{a} and the received data \mathbf{X} . It gets the information it needs to separate the M-space into the signal and noise subspaces from the spatial covariance matrix of \mathbf{X} , given by

$$\mathbf{R} = \left(\frac{1}{N-1} \right) \mathbf{X} \mathbf{X}^H \quad (7)$$

where the H superscript stands for Hermitian transpose (complex conjugate transpose). See Appendix A for a complete linear algebraic discussion of why MUSIC uses this covariance matrix and what its special properties are. For our purpose, the important thing to know is that since \mathbf{R} is Hermitian, its *eigenvalues* are real and non-negative. Also, its *eigenvectors* are orthogonal, so they form an *orthonormal basis* for the antenna array M-space. If we can separate these eigenvectors into those corresponding to the signals and those corresponding to the noise, we will then have orthogonal basis vectors for the signal subspace and the noise subspace, and these subspaces will be orthogonal to each other. Referring to Figure 4, the vectors \mathbf{e}_1 and \mathbf{e}_2 are the eigenvectors corresponding to the signal subspace. Notice that they define an orthogonal basis for the signal subspace. The vector $\mathbf{e}_{\min/\text{noise}}$ is the eigenvector corresponding to the noise subspace. Notice that it is orthogonal to the signal subspace defined by \mathbf{e}_1 and \mathbf{e}_2 .

Now the problem is determining which eigenvectors correspond to the signal subspace and which correspond to the noise subspace. From the results given in Appendix A, we also know that the eigenvectors corresponding to the noise subspace will have the smallest eigenvalues. Furthermore, these eigenvalues will all be equal. We can determine these eigenvalues by plotting the eigenvalues and finding where the eigenvalues stop decreasing and stay constant. The number of these minimum eigenvalues gives the dimension of the noise subspace, and their associated eigenvectors define the noise subspace. Similarly, the number of other eigenvalues (those larger than the minimum) gives the dimension of the signal subspace (as well as the actual number of signals), and their associated eigenvectors define the signal subspace. Thus, if N_{noise} is the number of noise eigenvalues, then $NSIG$, the number of signals, is given by:

$$NSIG = M - N_{\text{noise}} \quad (8)$$

The next step is to find the AOAs. This can be done in two ways. The first way to do this, mentioned above is to find the vectors in the array manifold that lie in the signal subspace. The points where the array manifold intersects the signal subspace would give the steering vectors, and hence the AOAs. The other way is to find the vectors in the array manifold that are *orthogonal* to the noise subspace. This is equivalent to the first method, since the noise and signal subspaces are orthogonal. We will use the second method because with the inaccuracies in our model and calculations, the array manifold might not actually intersect the signal subspace.

To find where the vectors of the array manifold are orthogonal to the noise subspace, we need to minimize the magnitude squared of the projection onto the noise subspace. To find the projection of each array manifold vector onto the noise subspace, we simply use the dot product:

$$\mathbf{p} = \mathbf{a}(\phi)^H \cdot \mathbf{E}_n$$

where \mathbf{E}_n is the noise subspace. We can then find the magnitude squared of this projection by calculating the dot product of the projection with itself:

$$\|\mathbf{p}\|^2 = \mathbf{p}^H \cdot \mathbf{p}$$

Notice that this formula is really the same as the Pythagorean Theorem in M dimensions. For example, if $\mathbf{p} = \begin{bmatrix} a \\ b \end{bmatrix}$ then $\|\mathbf{p}\|^2 = \mathbf{p}^H \cdot \mathbf{p} = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a^2 + b^2$. Clearly, this is just the Pythagorean Theorem in 2 dimensions as shown below:

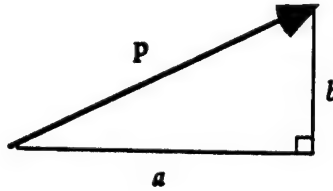


Figure 5: Pythagorean Theorem

Where the array manifold is orthogonal to the noise subspace, its projection will be zero. Thus, to find the steering vectors, we can compute the magnitude squared of the projections of all the array manifold vectors onto the noise subspace. We take the NSIG number of these with the smallest projections to be the steering vectors. In our case, we will use the inverse of this projection. If we plot the inverse of the projection of the array manifold onto the noise subspace versus AOA, we should then see peaks at the AOAs of the incoming signals. This plot is called the *MUSIC spectrum*. We can then use a peak-picking algorithm to select these peaks.

MATLAB Implementation

The MATLAB implementation of the MUSIC algorithm follows directly from the matrix representation given in the previous section. The function which implements the main part of the MUSIC algorithm is in the MATLAB function "MUSIC_AOAs." This function expects to be passed the data received at the antenna array, \mathbf{X} , and the array manifold, \mathbf{a} . It returns the AOAs of the signals, and the spatial covariance matrix \mathbf{R} . A separate function, "data_gen", was created by Lt Wintermyre to set up the

data matrix X for MUSIC_AOAs. MUSIC_AOAs calls two other functions, "pick_noise_eigs" and "pick_peaks." Pick_noise_eigs lets the user select the cutoff between the signal and noise eigenvalues. Pick_peaks returns the AOAs by locating the peaks in the MUSIC spectrum.

The MATLAB code for data_gen is in Appendix B.

```
function [AOAs,R] = MUSIC_AOAs(X,a)

% [AOAs,R] = MUSIC_AOAs(X,a)
%
% This function computes the angles of arrival of the signals in X using the
% MUSIC algorithm.
%
% INPUTS:
%       X = M by N matrix of data received by the antenna elements
%           (M = number of array elements, N = number of data sample
%           points)
%       a = array manifold calibration data
%           (M rows, number of columns depends on resolution of the
%           calibration data. We will assume the calibration data
%           represents 360 degrees.)
%
% OUTPUTS:
%       AOAs = vector of angles of arrival of the signals in X
%       R = spatial covariance matrix of X
%
% Stacy Fitzsimmons 8/15/94

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialize constants for MUSIC algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[M,N] = size(X);           % dimensions of X
[a_rows,a_cols] = size(a); % dimensions of a

a_res = 360/a_cols;        % resolution of array manifold in degrees

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NSIG (number of signals) Estimator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('Estimating NSIG_')
R = X*X'/(N-1);           % spatial covariance matrix
K = rank(R);              % rank of covariance matrix (should = M)

disp('Rank of spatial covariance matrix')
disp(K)

if K ~= M                 % data not full rank
    error('Aborting (data not full rank)')
end

[e_vecs,D] = eig(R);      % eigenvectors and eigenvalues of R (the eigenvalues
                          % are on the main diagonal of D)
e_vals = diag(D);         % extract vector of eigenvalues from diagonal of D
disp('Eigenvalues of spatial covariance matrix')
disp(e_vals)

cutoff = pick_noise_eigs(e_vals); % pick cutoff between signal and noise eigenvalues
```



```

N_noise = M - cutoff + 1;          % number of noise eigenvalues
NSIG = M - N_noise;                % estimation of the number of signals
disp('Estimate of number of signals')
disp(NSIG)

En = e_vecs(:, cutoff:M);          % noise eigenvector matrix
Es = e_vecs(:, 1:cutoff-1);        % signal eigenvector matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Signal AOA Estimator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

F = zeros(1,a_cols);               % allocate storage for F vector
for theta_index = 1:a_cols
    cur_a = a(:, theta_index);      % column of a at theta_index (the array
    % manifold for a specific value of theta)
    p = cur_a'*En;                  % compute projection of current array manifold
    % vector onto noise subspace
    F(theta_index) = p'*p;          % compute squared magnitude of projection

% This is an equivalent way to do the same thing
% F(theta_index) = (cur_a'*En) * (En'*cur_a);

end

F = real(F);                       % make sure F is real
F = 1./F;                           % invert so we see peaks at AOA's

theta_plot = -90+a_res:a_res:90; % x axis from -90 to 90 degrees
F_plot = [F((270/a_res)+1:length(F)), F(1:90/a_res)]; % extract sections of
% F vector that correspond to -90 to 90 degrees

semilogy(theta_plot,F_plot)        % plot the MUSIC spectrum (F)
title('MUSIC Spectrum')
xlabel('Angle of Arrival')
grid

AOAs = pick_peaks(F_plot);          % get the AOAs by picking the peaks in the spectrum
disp('AOAs of signals')
disp(AOAs)

```

Simulation Results

In order to test the MUSIC_AOAs function, we created script that generated a simulated collection scenario. The script is listed below:

```

% This script is a simulation of the MUSIC algorithm.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up parameters for data_gen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M = 3;                             % number of array elements
K = 2;                             % number of signals

max_snr = 10;                      % maximum SNR in dB (= 10*log(variance(signal)/variance(noise)))
% (assuming zero mean, so that mean_sqr = variance?)

```

```

AOAs = [30, 80];    % angles of arrival of signals (in degrees; should be between
                    % -90 and 90; 0 degrees corresponds to broadside)

Amp = [1, 1];       % amplitudes of signals (1 is maximum)

fs = 8000;          % sampling rate
N = fs/2;           % number of sample points
array_res = 0.5;    % array manifold resolution in degrees
f_loc_osc = 100000000; % frequency of array receiver local oscillator
                    % (100 MHz)

fc = [2000, 1900];  % carrier frequencies of FM signals (note that we're
                    % using lower frequencies so we can hear the resulting
                    % FM signals)

fdev = [100, 100];  % peak frequency deviations of signals, in Hz

% Generate our modulating signals. In this case, we are using simple tones,
% but in general, the modulating signals could be any signals (such as speech)

fm = [200, 500];    % frequencies of modulating tones
t = 0:1/fs:(N-1)/fs; % generate time vector
m = zeros(K,N);      % allocate storage
for ind = 1:K        % generate tones
    m(ind,:) = cos(2*pi*fm(ind)*t);
end

% Generate the data matrix and array calibration data
[X,a] = data_gen(M,K,m,fc,fdev,fs,AOAs,max_snr,f_loc_osc,Amp,array_res);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Call MUSIC_AOAs to get signal AOAs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Est_AOAs, R] = MUSIC_AOAs(X, a);

```

In this particular case, we had an antenna array with 3 elements, 2 signals (simple tones at frequencies of 200 Hz and 500 Hz) with AOAs of 30° and 80° respectively. When the script calls MUSIC_AOAs, it presents the user with a graph of the eigenvalues, so that the user can select the cutoff between the noise and signal eigenvalues, as shown on the next page:

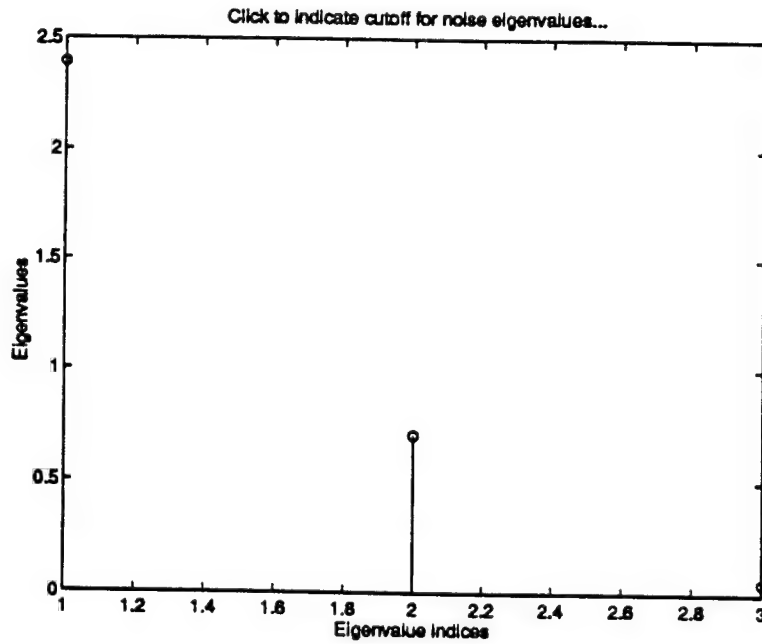


Figure 6: Selecting Eigenvalue Cutoff

From the graph it is quite clear that the cutoff should be between the 2nd and 3rd eigenvalues. After clicking to select the cutoff, MUSIC_AOAs computes the MUSIC spectrum and returns the AOAs. The MUSIC spectrum for this example is shown in Figure 7. It clearly shows peaks at 30° and 80°, which are in fact the AOAs of the original signals.

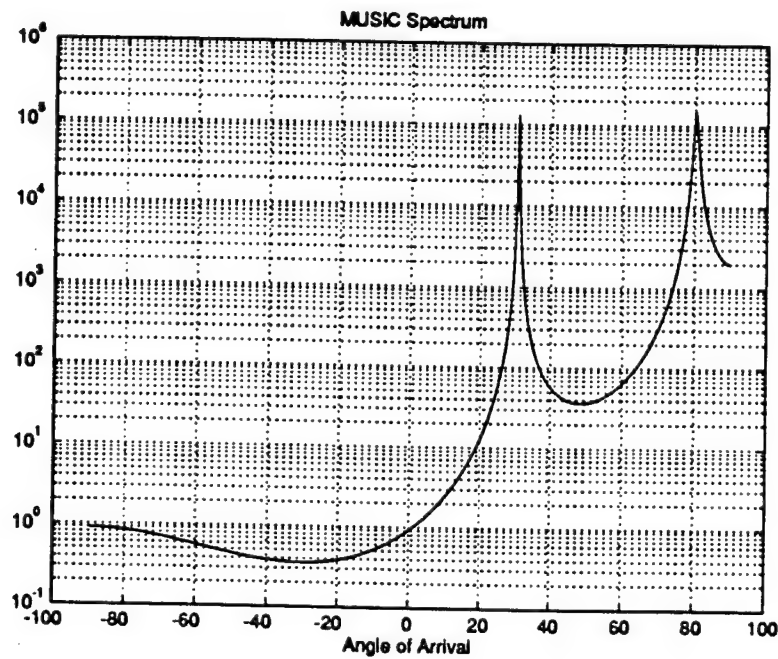


Figure 7: MUSIC Spectrum for 2-Signal Simulation

Conclusion

Since separating signals to process them can be difficult, especially on an airborne platform, it is necessary to find alternative ways to the conventional processing methods that use frequency as their main factor in the separation. Using space instead of frequency is just as efficient and it eliminates the problem of separating several signals that share the same frequency bandwidth. The MUSIC (Multiple Signal Classification) algorithm can be conveniently implemented in MATLAB and used to locate specific signals so they can later be separated and processed.

The MUSIC implementation described here (along with its other associated MATLAB functions) will be used in IRAA as the foundation for an in-house capability to simulate spatial processing algorithms.

References

Leon, Steven J. Linear Algebra with Applications: Second Edition. New York: Macmillan Publishing Company, 1986.

Oppenheim, Alan V., Alan S. Willsky and Ian T. Young. Signals and Systems. New Jersey: Prentice-Hall, Inc. , 1983.

Schmidt, Ralph Otto. A Signal Subspace Approach to Multiple Emitter Location and Estimation. Michigan: University Microfilms International Dissertation Information Service, 1982.

Technical Consultation

1Lt. James Wintermyre - MATLAB, signal processing, MUSIC algorithm, final report consultation

Rollie Holman - MUSIC algorithm description, Appendix A, final report consultation

Michael Weir - Linear algebra, signal processing, final report consultation

Appendix A: A Student MUSIC Algorithm as an Application of Linear Algebra with Geometric Interpretation

This appendix includes a description of the problem for the summer apprenticeship program, as well as the linear algebra behind the MUSIC algorithm. It was written by Rollie Holman to be included as a supplement to the final report.

A.1 INTRODUCTION

Presented here is a student problem for implementation of a Multiple Signal Classification (MUSIC) algorithm. A MUSIC implementation serves as an instructive project requiring a college level knowledge of linear algebra and facility in the use of MATLAB. This appendix presents the linear algebra rationale for the MUSIC algorithm for the student implementation. Where relevant, geometric interpretation of the signal/vector processing steps is stressed. This write up also places the student MUSIC implementation in a context of application and algorithm maturity.

The principal references are:

Schmidt, Ralph Otto, *A signal subspace approach to multiple emitter location and spectral estimation*, Ph.D. Dissertation, Stanford University, 1982, TK5102.5.S35 1982.

Strang, Gilbert, *Linear algebra and its applications* ; San Diego : Harcourt,

Brace, Jovanovich, Publishers, c1988. QA184.S8 1988; 3rd ed.; xii, 505 p. : ill. ; 24 cm.;

Many college texts on Linear Algebra may serve to substitute or augment the later reference.

MUSIC as implemented here, applies to data collected by an array of sensor elements where the solution sought is the direction of arrival(s) (DOAs) of one or more signals incident on an array. The signals may be of many types, such as communication, acoustic, and seismic signals, where the array sensor elements are appropriate transducers for the application signal type. For communication signals the array consists of antenna elements. In all applications the array geometry is designed to accomplish a suitable spatial sampling of multiple signal wavefronts, at any snapshot in time.

The implementation here assumes a planar geometry, that is the signal source locations and sensor array elements all lie in the same plane. The solution DOAs are azimuthal angles in degrees, relative to the sensor array orientation. For application to communication signals, the array is modeled as an arrangement of dipole elements oriented perpendicular to the plane (The center of the dipoles lay within the plane). In all cases an array response calibration is provided. The array response calibration, termed the array manifold, gives the array steady-state response as a vector for a single incident emitter at a given DOA. The vector dimensional components are the outputs of the individual array elements. In principal, the array manifold is a continuum of array response vectors corresponding to a continuum of possible DOAs, that is 0 to 360 degrees. In practice, the provided array manifold is a matrix of column array response vectors representing a closely spaced sampling of a continuous array manifold.

In general, the array manifold is a function of the signal frequencies. For the student MUSIC implementation, the signal bandwidths are constrained to be much less than the array bandwidth. This means that the phase differences observed between the individual array elements for snapshot data vectors result for all practical purposes from only the spatial sampling and not from signal modulations. Thus a single frequency array manifold serves. The carrier frequencies of incident signals are assumed to be within the defined signal frequency channel.

The array calibration or manifold can be derived from an analytical model of a subject array. For the student problem, a linear antenna array with uniformly spaced dipole elements is modeled. Often in practice the array manifold results from a measurement calibration process. Calibration by measurement is necessary because of "real-world" complexities that defy accurate modeling. Examples are array imperfections, mutual coupling between array elements, and near-field scattering effects due to the array platform and support structures. It is important to point out that the student MUSIC algorithm can work against real-life array collected data as well as simulated data. Also, it matters not whether the provided array manifold data results from analytic modeling or calibration measurements.

For the student MUSIC algorithm, the number of incident signals (K) is less than the number of array elements (M). Furthermore, the modulations of the individual signals are such that they are uncorrelated. In real situations, correlated incident waveforms can result from discrete multipath. However, this is beyond the scope of a student MUSIC implementation. The student MUSIC algorithm can process data from any size array and number of signals, as long as the number of signals is less than the array size. However, the example data set was generated for an array of three elements. The use of three dimensions makes easier the geometric interpretation of the linear algebra vector processing steps. Most people have difficulty visualizing in dimensions greater than three.

The student MUSIC algorithm assumes that the noise experienced by the individual array elements is independent. Such noise is said to be spatially white. In practice noise can be spatially colored. In communications applications, an example of spatially colored noise is urban/industrial electromagnetic noise. The student MUSIC algorithm can be extended to deal with colored spatial noise. Other potential extensions include 1) the estimation of additional parameters (e.g., elevation as well as azimuthal DOA); 2) correlated signals due to discrete multipath; and 3) signal separation beamforming and extraction of individual signals. In short, the student MUSIC implementation is a serious signal processing building block, that can be extended in its capability.

MUSIC is a "subspace" algorithm. The procedure is to partition the M -space into two orthogonal complementary subspaces, a signal and a noise subspace. The solution signal directional vectors are the K array manifold vectors that lie within or closest to the signal subspace. A suitable number, N of collected snapshot data vectors are required to provide an adequate estimate of the spatial covariance matrix. Computed eigenvalues of the covariance matrix are used to determine the number of signals, K . The corresponding computed eigenvectors, returned as a matrix of column vectors, are an orthonormal basis set for the M -space. Knowledge of the number of signals, K allows the column space of eigenvectors to be partitioned into the complementary subspaces. A least mean squares projection approach is then used to identify the K array manifold vectors that fit best to the signal subspace. These are the estimates of the signal directional vectors, which in turn identify the DOAs.

There are other applications amenable to the MUSIC or subspace approach. MUSIC is useful for spectral analysis of signals that are a mix of sinusoids and noise. Here, the data results from a set of measurements from a tapped delay line. The application is to identify the element signal frequencies, amplitudes and phases.

In a broader context, MUSIC relates to factor analysis, a discipline which is applied to problems in many fields, including psychology and economics. Here, the researcher seeks to identify a subset of most significant factors or causes from a greater set (see Mardia, K.V., Kent, J.T., Bibby, J.M., *Multivariate Analysis*, Academic Press, 1979).

A.2 DATA MODEL

A.2.1 Measurement vectors

The data is in the form of a collection of measurement vectors, where each measurement vector is the observed array output at a snapshot in time. The dimensions of a snapshot vector are outputs of the individual array elements, of number M . The set of all conceivable measurement vectors exist in the M dimensional space, C^M , as do the array manifold vectors, $a(\alpha)$.

Any snapshot measurement vector can be expressed as the sum of two terms, a deterministic term plus a noise term,

$$x(t) = A(\alpha) s(t) + n(t)$$

where

$$s(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_K(t) \end{bmatrix}$$

is a column vector of complex amplitudes of the K signals at time t . The matrix,

$$A(\alpha) = [a(\alpha_1) \mid a(\alpha_2) \mid \dots \mid a(\alpha_K)]$$

is made up of column vectors which are respectively the signal directional vectors (specific array manifold vectors) associated with the K incident signals. The noise vector,

$$n(t) = \begin{bmatrix} n_1(t) \\ n_2(t) \\ \vdots \\ n_M(t) \end{bmatrix}$$

is made up of the additive noise samples at each of the array elements. The noise is Gaussian and uncorrelated between elements.

A.2.2 Multiple Snapshot Collections

The data sets provided are multiple snapshot collections of N column vectors. N needs to be large enough to provide for suitable estimates of a spatial covariance matrix. A collection is expressed as

$$X = A(\alpha) S + V$$

where X is a M by N matrix of N column data snapshot vectors. S is a K by N matrix of column vectors which are multi-signal samples at the N snapshot times. It is noted that the matrix product $A(\alpha) S$ results in a matrix that is M by N . V is a M by N matrix of column vectors which are the N snapshots of the M -dimensional noise samples.

A.3 MEASUREMENT COVARIANCE MATRIX

The estimated spatial covariance matrix, assuming zero-mean signal and noise processes, is

$$R_{XX} = \frac{1}{N-1} \sum_{n=1}^N \{x_n x_n^H\} = \frac{1}{N-1} X X^H$$

Given an adequate data sample size, the spatial covariance matrix can be expressed as the sum of two terms, that is

$$R_{XX} = W + \sigma^2 I$$

where W is the signal spatial covariance term, $\sigma^2 I$ is the noise covariance term, and I is the identity matrix. The noise covariance matrix is diagonal, as a result of uncorrelated Gaussian noise experienced by the array elements. Furthermore, the noise power at each of the array elements is equal to σ^2 . Thus, the noise covariance matrix is of rank M ; it is of full rank.

W , the signal spatial covariance matrix, is modeled as deterministic. This matrix is of rank K , the number of uncorrelated incident signals. W is not of full rank and thus is singular. R_{XX} is of full rank as a result of the noise matrix term.

A.4 EIGENANALYSIS

The eigenvalues of the matrix R_{XX} can in principal be obtained by solving

$$|R_{XX} - \lambda I| = 0$$

for M roots (eigenvalues) $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M \geq 0$. The fact that the eigenvalues are non-negative and real follows from the fact that the covariance matrix is Hermitian. Equivalently, we must satisfy

$$R_{XX} E = E \Lambda$$

where $\Lambda = \text{diag} \{ \lambda_1 \lambda_2 \dots \lambda_M \}$ and $E = [e_1 | e_2 | \dots | e_M]$ is the column matrix of eigenvectors. Since R_{XX} is a Hermitian matrix, the set of all its eigenvectors provide an orthonormal basis for vector space \mathbb{C}^M .

The problem now is that of partitioning this column space, E in to two complementary orthogonal subspaces, one associated with the signal and the other with the noise. In so doing the number of signals, $NSIG$ equal to K , is defined. We start by noting that W is singular (not of full rank), so that its determinant is equal to zero, i.e.,

$$|W| = |R_{XX} - \sigma^2 I| = 0$$

We also note that from the definition of eigenvalues of R_{XX} (equation repeated from above) that

$$|R_{XX} - \lambda I| = 0$$

Thus, at least one of the eigenvalues, $\lambda = \sigma^2$, but which one? The Hermitian covariance matrix is positive-definite, such that its eigenvalues are non-negative real numbers. This fact constrains

$$\lambda_{\min} = \lambda_M = \sigma^2$$

If this were not the case, then there may be some possible data collects where the minimum value, λ_{\min} is negative. But this can not be.

The eigenvector corresponding to λ_{\min} (or vectors) is orthogonal to all other eigenvectors and hence exists within the nullspace of the signal subspace. Thus, $W e = 0$ for all eigenvectors, e which are orthogonal to the signal subspace. By definition then,

$$R_{XX} e = \sigma^2 I e = \lambda e$$

for the noise associated eigenvalues and their corresponding eigenvectors. From this it is stated that all the $M-K$ eigenvalues corresponding to the noise subspace are equal to σ^2 . The value of $\lambda_{\min} = \sigma^2$ is then the estimate of the noise power or variance experienced by the individual array elements.

Since these are the $M-K$ smallest eigenvalues of the spatial covariance matrix, the corresponding eigenvectors are the set of column vectors that are the orthonormal basis vector set for the noise subspace. The noise subspace then is,

$$E_N = [e_{K+1} | e_{K+2} | \dots | e_M]$$

spanned by the eigenvectors corresponding to the eigenvalues, $\lambda_{K+1} \geq \lambda_{K+2} \geq \dots \geq \lambda_M \geq 0$. The signal subspace then associates with the remaining eigenvectors, that is

$$E_S = [e_1 | e_2 | \dots | e_K]$$

spanned by the eigenvectors corresponding to the eigenvalues, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_K \geq 0$. The two subspaces E_N and E_S are orthogonal and complementary.

To provide further enlightenment, the K eigenvalues γ_k (all positive real numbers) of W are defined as

$$|W - \gamma_k I| = |R_{XX} - (\gamma_k + \sigma^2)I| = 0$$

Thus, $\gamma_k + \sigma^2$ are the K eigenvalues associated with the signal subspace. It is observed that they are all larger than the eigenvalues associated with the noise subspace, that is larger than σ^2 .

A.5 ESTIMATION OF NUMBER OF SIGNALS (NSIG) AND DETERMINATION OF THE SIGNAL AND NOISE SUBSPACES

The fact that the $M-K$ eigenvalues of the spatial covariance matrix that associate with the noise subspace are equal to the minimum eigenvalue and that the eigenvalues associated with the signal subspace are all greater in value to the noise subspace values is the basis for determining K , that is the number of uncorrelated signals termed NSIG. In the implemented MUSIC algorithm, NSIG is determined by user interactive interpretation of a presented graph of the M eigenvalues of the spatial covariance matrix.

For instructive purposes, the interactive approach is suggested for the NSIG determination. An extension to the student MUSIC would be an automatic thresholding means for estimation of NSIG.

The Information Superhighway: Still Under Construction

David W. Gurecki

Rome Catholic High School
800 Cypress St.
Rome, NY 13440

Final Report for:
High School Apprenticeship Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC
and
Rome Laboratory

August 1994

The Information Superhighway: Still Under Construction

David W. Gurecki
Rome Catholic High School

Abstract

Advances in multimedia technology will greatly affect the future of computers. The "Information Superhighway" is a term referring to the exchange of almost any type of information over a vast, world-wide super-network of computers. This network exists today (commonly referred to as the Internet), and the technology to interface video, audio, imagery, and data into computers also exists today. However, much of this technology is still in the development stage, and there are problems that have to be resolved.

This report describes some of these new areas of multimedia and communicating technology experimented with, the problems which can arise, and some possible solutions to these problems. It covers two applications that were developed during this period ("WEB-Link" and "ScanRes").

The Information Superhighway: Still Under Construction

David W. Gurecki

Introduction

Recent advances in computer performance, video and audio processing, and telecommunications have expanded the utility of computers. With Intel Pentium technology, PC computers are now capable of speeds up to 100 MHz. Video and sound are now being interfaced with computers. Scanning technology allows any printed image to be digitized into raster format for use in computing. Most importantly, the ability to share all this multimedia information has expanded exponentially through the growth of the Internet, a mass networking of billions of computers around the world. However, at the current time, this "Information Superhighway" has several barriers which must be overcome. Much of the technology and software available today has to be optimized and experimented with to determine its usability.

The issues addressed for this project include an explanation of a number of these barriers:

1. the expansion of an existing office messaging system into a new, versatile information exchange environment called the World Wide Web;
2. determining video recording and playback factors to optimize performance;
3. problems associated with the interdependability of hardware and software from several different developers;
4. the viability of video teleconferencing, and which software is best for what application;
5. determining the best resolution for scanned images to minimize distortions.

These problems, among many others, have to be resolved before the mass public can effectively benefit from these advances. Technology must be made to work efficiently and consistently. Some solutions were found to these problems identified above, but many other questions still have to be answered before the Information Superhighway becomes a working reality.

The World Wide Web Information Exchange Environment

Traditionally, information has primarily been exchanged over the Internet through several text-based utilities (i.e., File Transfer Protocol (FTP) and Gopher, a menu-based interface). These both allow transmission of files to almost any location on the Internet, but are somewhat user-unfriendly since the user must employ multiple steps to see or listen to multimedia information.

The World Wide Web is an increasingly popular information exchange environment. Unlike FTP or Gopher, it operates in a graphics environment. It utilizes hyper-text linked files to display text, graphics, sound and animations from anywhere over the Internet. The user can view graphics and hear sound simply by selecting a link object (a labeled area on the screen that appears in blue). This is a much simpler, user-friendly method for selecting and transferring multimedia information over the Internet.

The advantages of the WEB environment and its increased usage at Rome Laboratory led to the idea to include the Video Bulletin Board¹ sequences in the WEB environment. This complicated task would involve capturing the screens displayed, converting them to a WEB-compatible format, and transferring them to a WEB server that automatically puts them on-line. The versatility of the WEB environment makes it possible for graphics to be easily transferred to users' terminals, and made this endeavor a practical undertaking. This entire process would have to be automated for ease of use and practical application. A special software application called WEBLink was developed for this purpose. The following is a description of this development effort.

The best approach to adapting the Video Bulletin Board to the WEB environment was to incorporate both image files (i.e., PCX files) and text exactly as they appear on the TV screens into the WEB. Consequently, a procedure to capture the graphic screens was investigated. Although there are many external utilities that can capture graphics screens and perform the function adequately, they could not be included in an automated process since a keypress is required. Further research indicated that the IVV² software itself has built-in routines which can capture the screen to a

¹ The Rome Laboratory Video Bulletin Board originated as a project to improve communication of news and events within a building. Initiated in 1993, this project involves the displaying of video messages, or "sequences" on TV monitors throughout the workplace.

² IVV is the Interactive Virtual Video™ Software by V_Graph, Inc. used to display and create sequences for the Video Bulletin Board.

PCX file. When the IVV command processor encounters a "SavePic" command in a sequence file, it captures the entire graphic screen and saves it to a PCX-format image file on disk.

The SavePic command was tested, and found to work properly with some screens and return an error with others. After contacting the author of the software, the problem was discovered to be an incompatibility with the video board. The capture worked on 16-color (4 bit) screens, but not on 256-color (16 bit) screens. Therefore, the system had to be reconfigured with a new video board that worked correctly with the IVV software. This is an excellent example of a type of problem when integrating software and hardware from different developers onto the Information Superhighway computer systems.

Once a capturing method was determined, the RL Scheduler (the program which schedules the messages) had to be modified for this effort. Previously, the RL Scheduler played each sequence once, and then looped back to the beginning. However, for the automated process the Scheduler was modified to display each sequence only once, and then quit.

The IVV SavePic command saves the screen into PCX format. Unfortunately, PCX is not the ideal format for viewing under the WEB environment, since Mosaic (a WEB interface program) was not designed to recognize PCX files. Mosaic can, however, recognize images in the GIF image format. Since it is possible to convert image files from PCX to GIF, a conversion program was acquired. The CVTGIF 1.5 utility, by John Bridges, was tested and proved to be sufficient for converting PCX files into GIF format.

Once the files are created and converted, they have to be transferred to a WEB Server. This is accomplished by using the FTP program to send all the captured screens from an IBM computer to a UNIX server. This program can be set up to read in commands from a textfile. Since the files that need to be transferred will be different for each session, this textfile can be modified each time to reflect the appropriate filenames.

In the schedule file, there is a description field containing a short title message about each sequence. When the automated capture process runs, a list is created which contains (1) the description (from the schedule file) of each sequence, and (2) which GIF files were created from each sequence. This list is also sent to the remote UNIX machine with the GIF files. There, it is used to create a hypertext file. For each sequence, the title is listed and the screens that

were captured from that sequence are listed below it. Using Mosaic, the users can select whichever screen they wish to view, and the corresponding GIF file will be sent to their computer and displayed.

A special program segment was developed for the UNIX machine to scan the GIF file directory once every fifteen minutes. If it detects the presence of any GIF files, it will move them into the WEB directory where users can download them, and revise the listing to reflect the new GIF files that were uploaded.

This WEBLink project was completed successfully and is now operational. It takes full advantage of the new WEB environment's ability to transfer images quickly and easily to any user viewing the hypertext file. It's user-friendliness and ease of use makes this environment likely to continue to be used in the future for transferring multimedia information on a global basis.

Video Recording and Playback Experimentation

Technology exists today that allows interfacing a video camera to a computer and recording digitized animation sequences from such a video source. When recording, there are many configurable options that will considerably effect the quality, smoothness, and performance of the recorded video clip. The biggest obstacle to recording video is that as the video is being received by the computer, it must be promptly written to disk. Generally, the less information that is written to disk, the more video frames can be sampled per fixed length of time. Several factors affect this disk transfer rate (the rate at which data is written to disk), and thus affect the frame rate (the rate at which the computer can capture video frames). These factors include: the speed of the microprocessor, the dimensions of the window in which the video is being recorded, the presence or non-presence of sound, the use of non-use of compression when writing the file to disk, and the length of the video. The following illustrates how the above factors affect the disk rate and frame rate:

1. A faster microprocessor will allow for greater speed in getting the data to the disk itself, thus increasing the disk rate.

2. A large recording window slows down the disk rate because there is more picture data that must be written to disk. If the system is busy writing to disk, it can't be sampling image data at the same time, so a large recording window limits the sampling rate. Conversely, a small recording window means less image data is recorded to disk, so

the system can spend more time sampling the incoming video. It can capture more frames per second since it can write more frames to disk in less time.

If the sampling rate is too high for any size window, the computer will lose frames from the video since it has to finish writing to disk while the video is still continuing. These factors create a threshold which had to be determined for each of three sizes of recording windows. (see below)

3. Compressing the video information as it is being written to disk decreases the amount of information that needs to be written. Using an algorithm such as Intel Indeo decreases the information stored for each frame, so more frames can be sampled per second and the frame rate can be higher.

4. The presence of audio decreases the rate at which information can be written to disk because twice as much information needs to be stored, as opposed to only video or only audio information.

5. Systems implementing a disk caching program have the ability to hold information in memory and write it to disk at a later time when the system isn't as busy. This speeds up the disk rate on a short-term basis, because after the buffer is filled, the disk rate goes back to normal again. For this reason, the length of the video becomes a factor. Very brief video clips may be able to have higher frame sampling rates because the disk rate is still very high. However, even when disk caching is employed, a long video clip at too high a frame rate will still cause parts of the video or sound to be lost later on in the clip.

In general, a faster frame rate will create a much smoother video image. However, if the computer is trying to sample sound and video at a rate faster than it can store it, video frames and audio will be lost, and the video will not be presentable. Experiments were done to determine the best possible frame rates for each of three window sizes in order that no frames or audio would be lost. Many video clips were recorded at varying lengths, frame rates, and window sizes to determine both the relationships stated above, and the frame rate threshold at which it is practical to record video clips.

These experiments used the Intel Video for Windows software and an Intel Indeo video capture board. The compression method used was Intel Indeo R3.0. Each clip was recorded with a Pentium-60, compression on, with both audio and video input. The table below shows the maximum frame rates at which no video and no sound was lost for ALL video clips from 5-30 seconds long.

Resolution	Maximum Frame Rate
320x240 pixels	8 frames/second
240x180 pixels	14 frames/second
160x120 pixels	29 frames/second

Note that these values cannot be completely accurate. These values are the threshold at which it generally becomes possible to record a video clip without frame and audio loss. A higher frame rate may be achieved if the video clip is only 5-10 seconds long without any frame loss. However, some video clips may not be as compressible as others, and this can push the threshold to a lower value. If video or audio loss does occur at one of these frame rates, decreasing the frame rate may be able to solve the problem.

Even after a video clip has been correctly recorded, there may sometimes be difficulty playing it back as it was recorded. Sometimes sound can lose synchronization with the video, depending on how fast the computer can read data from a disk and process the playback video and sound information. If the system processor isn't fast enough to load and display 1 frame of both video and sound information at the same rate that it was recorded, there may be sound loss, jumpiness, or loss in synchronization with the video. One way to compensate for this condition is to change the audio interleave from every frame to every 5 frames (or higher). This causes the audio for the next 5 frames to be interleaved in a single frame. Hence, the system doesn't have to worry about audio for another 5 frames, and can devote all its resources to displaying the video correctly.

Despite all this optimization and operation on a top level PC, there is an inherent limit to the maximum quality that can be attained. In the future, as computer systems become faster, this limit will become higher and higher until very high quality video will be attainable.

Memory Conflicts and Optimization under MS-DOS

In order to support a full range of capabilities such as video capture, sound, scanning, Network connections, and CD-ROM support, PC computers must have numerous expansion boards installed in the system. In addition, the manufacturers of these components also require separate resident software device driver modules (TSRs³) to interface the hardware to the system. The MS-DOS environment is limited as to the amount of directly addressable random access memory (RAM) it has for TSR storage. While a PC may have 16 MB of RAM, much less than 1 MB is actually available for TSRs. There also needs to be some space left for regular programs. Therefore, as the number of multimedia system features increase, the amount of RAM memory required to properly operate them also increases. Since there is a very limited space, having many multimedia features on a system can be as much of a hindrance as a benefit.

The MS-DOS memory map is defined as follows:

0-640 k	Executable programs and TSRs
640-1024 k	System BIOS and Video ROM
1024-max k	Extended memory

Normally, all TSR programs loaded would go in the 640k region. In some multimedia systems, that has left memory available for regular programs at less than 300k, which is not enough to run some programs. The 640-1024k region is used to store BIOS and Video ROM information, as well as drivers for other system components. Occasionally, between some of these system regions, there is some free space available. The amount will vary with computers, and there may be up to 200k empty space or more. Fortunately, there are utilities, both within the latest versions of DOS and from third party companies, that can make use of these empty blocks of memory and use them to store the TSR driver programs.

³ TSR - "Terminate and Stay Resident."

After having optimized several systems in the workplace, it was found that this process can become very complicated. TSR programs tend to take up one size in memory when they execute, and then shrink to smaller size when it stays resident. For example, if there is a 20k block of free memory, and a 18k TSR needs to be loaded, it may seem obvious that it would fit in the 20k block. However, this TSR's initial size is 50k, which is larger than the available memory block. Hence, it is not possible to fit the TSR in that particular memory region. Since this is an example, situations like this have occurred very often, and it becomes a very time-consuming process to get all the required TSRs loaded in the best fit they can.

In some cases, all the TSRs can be loaded into UMBs (Upper Memory Blocks) in the 640-1024k region of memory. However, it is more typical that only a few of the TSRs fit in UMBs and the rest have to fit in regular system memory, which greatly decreases the memory available to run programs. In these cases, the system is better off than it would be without these memory managers. However, it may still be incapable of running some programs that it otherwise could run without many multimedia features. This dilemma supports the postulate that advancing technology for computers makes them worse instead of improving them. This is certainly not the intent of developers of multimedia components, nor the desire of home computer owners and prospective multimedia customers.

Currently, MS-DOS is the most widely used operating system among home computer users. However, there are alternatives to this operating environment. For example, Windows NT from Microsoft, OS/2 from IBM, and the upcoming Chicago from Microsoft are all PC operating systems which are not limited to under 1 MB storage for resident drivers. If these operating systems become more widely used in the future, as they may, the problems caused by the current memory restrictions of DOS will cease to exist.

Hopefully, these new operating systems will be capable of spreading throughout personal computer owners, and multimedia will become a viable computer upgrade option. When this happens, the Information Superhighway can more quickly extend into the homes of PC owners.

Video Teleconferencing

One relatively new capability which has become viable as a result of advances in technology is Video Teleconferencing. This application makes use of video capture, audio capture, and high-speed network communications to transmit live video and audio to another location.

Video teleconferencing has many uses. It can be used by doctors at home to make a quick decision in an emergency situation, to view x-rays, or make a diagnosis on a medical condition. It can be used by businesses to conference with distant branches, thus saving travel expenses and commuting time. Schools and universities will be able to take advantage of professors teaching specialized topics or access libraries of information in many remote locations. It is also useful simply for communicating between people, just as the telephones are used today.

There is a difference in the amount of quality required for different tasks. For example, a doctor looking at an x-ray will require more resolution than simply two relatives talking with each other. At the present time, there are several different applications which provide differing levels of quality.

One of these video image exchange systems was installed and evaluated during this work period. This program, called "CU-SeeMe," is being developed at Cornell University by Tim Dorcey, Steve Edgar and Richard Kennerly. This program runs over the Internet and has been written for Macintosh and Microsoft Windows. It currently has video and sound sending and receiving capabilities for the Macintosh, but only video send and receive for Microsoft Windows. The requirements for sending video are a camera and video capture board, and a connection to the Internet. To receive video, only a connection to the Internet is required. When receiving a video picture of another person, the image data appears in a 4-bit grayscale window 160x120 pixels in dimension. CU-SeeMe can do peer-to-peer teleconferencing, as well as conferencing with up to 8 other people.

Since the picture is very small and isn't in color, the quality may not be suitable for some higher-level applications, such as the doctor example. It is more practical for private use at home, however, it only runs on the Internet, and most people have either a very slow or no Internet connections in their home. This program should gain quick adoption for several reasons. It is still one of the first widely obtainable programs to emerge, since it can be downloaded from anywhere on the Internet and is public domain software. The video camera only costs about \$500 to

implement. Thus, although this program can't be used for high resolution tasks, it still may be adequate for business use or eventually private use as a tele-video system in place of the telephone.

Another teleconferencing program reviewed is called ProShare, which is being developed by Intel Corporation. This program currently runs only in the Microsoft Windows environment, and requires a specific communications media called Integrated System Digital Network (ISDN) to communicate with other ProShare systems. This is a much more versatile application. It allows for sharing not only color video and sound, but also other software applications. A remote user can see and operate the same programs as the local user. It provides much more opportunity for sharing data and productivity. This capability could be used by doctors and businesses, but because of the expense, it may be out of range for private users.

As stated earlier, the teleconferencing sessions are currently limited to computers with ISDN connections. ISDN is not very prevalent, thus limiting its wide-range use and making it expensive to operate. In the future, this program will support Ethernet LANs and eventually regular phone line connections. At the present time, however, CU-SeeMe provides more connectability, even though it has less capability.

These are only two examples of the many emerging systems, but they address some of the trade-offs that will occur in the video teleconferencing environment. More expensive software packages will be able to provide high quality and more features, but these won't be necessary for all people. Many users will be willing to settle for less quality and less expense. This cost/benefit trade-off will most likely prevail until higher-end software can reach affordability to the everyday user and compete with the lesser quality programs. This will occur when there is a larger customer base and video teleconferencing on PCs becomes of age.

Scanner resolutions

When scanning images into a computer using any type of digital image scanner, an important factor in producing a quality digital image is the resolution. If the resolution is unnecessarily high, processing the image will take a large amount of time and impede flexible handling within applications. However, a low resolution will result in a loss in quality of the image. Also, if a scanned image has to be resized using software functions, distortion will often occur and there can be severe loss of picture quality. The best solution is to scan the image at the best possible resolution for

the desired image size on the target media. There are formulas which can be used to calculate this value. A special Windows application called ScanRes was developed to provide a simple interface for calculating the resolution at which to scan an image.

ScanRes runs in the Microsoft Windows 3.1 (or later) environment, and works best with a mouse. It was developed with Microsoft Visual C++ for Windows 1.0 and takes approximately 1 MB of disk space. It can be executed by adding it as an item into the Program manager, selecting "Run" from the "File" menu and entering the full path and name to ScanRes, or double-clicking on the SCANRES.EXE file from within the Windows File Manager.

When ScanRes loads, a menu bar with four items appears. The "File" submenu only contains the "Exit" command to exit the program. The "Picture" submenu contains the options "Image" and "Target". "Image" allows the user to give the dimensions of the picture being scanned in. Selecting "Target" will allow the user to configure the target properties where the scanned image will be put. The "Calculate" option on the menu bar will compute the optimum scanning resolution for the entered data, and "About" gives information about the program. Following is an illustration of the dialog box that appears when "Image" is selected from the "Picture" Menu:

Scanned Image Properties

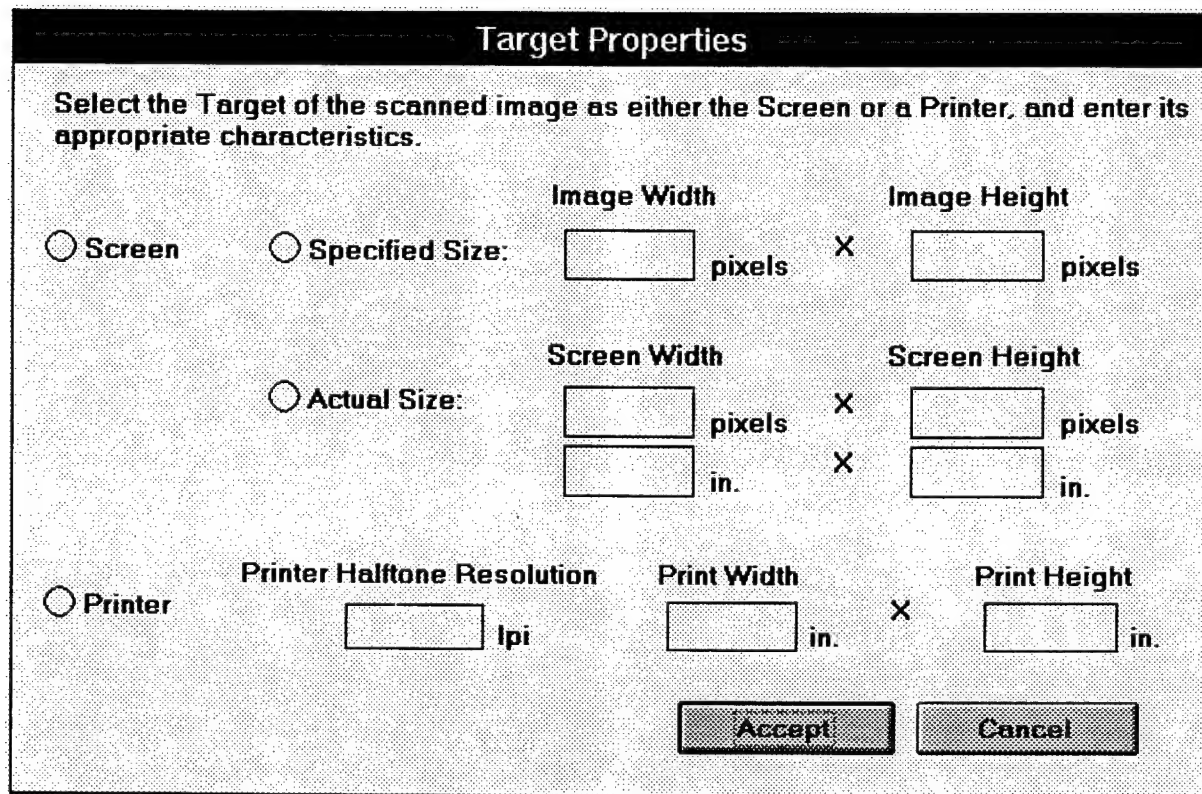
Enter the dimensions of the scanned image in the boxes below.

Scan Image Width Scan Image Height

0. in. X 0. in.

This window contains two boxes for entering data ("Scan Image Width" and "Scan Image Height") and two buttons (Accept and Cancel). The user must type the width of the Scanned image in the "Scan Image Width" box and the image's height in the "Scan Image Height" box, both in inches. Decimal values are accepted. After those values are selected, the user may select "Accept" to store the values in memory, or "Cancel" to close this window without saving the

values to memory. Following is an illustration of the dialog box that appears when "Target" is selected from the "Picture" menu:



The dialog box is titled "Target Properties". It contains a text box with the instruction: "Select the Target of the scanned image as either the Screen or a Printer, and enter its appropriate characteristics." Below this, there are three main sections. The first section is for "Screen" target, with a radio button labeled "Screen". It has two sub-sections: "Specified Size" with fields for "Image Width" (pixels) and "Image Height" (pixels), and "Actual Size" with fields for "Screen Width" (pixels and inches) and "Screen Height" (pixels and inches). The second section is for "Printer" target, with a radio button labeled "Printer". It has fields for "Printer Halftone Resolution" (lpi), "Print Width" (inches), and "Print Height" (inches). At the bottom right are "Accept" and "Cancel" buttons.

Target Properties

Select the Target of the scanned image as either the Screen or a Printer, and enter its appropriate characteristics.

☐ **Screen**

☐ **Specified Size:**

Image Width pixels X **Image Height** pixels

☐ **Actual Size:**

Screen Width pixels X **Screen Height** pixels
 in. X in.

☐ **Printer**

Printer Halftone Resolution lpi

Print Width in. X **Print Height** in.

Accept **Cancel**

In this dialog box, the user must select the target device and enter the properties of the target image. Each circle that appears to the left of the labels "Screen", "Printer", "Specified Size", and "Actual Size" can become filled when one is selected by the user. The user can select either Screen or Printer as a target device. (If Screen is selected and the user selects Printer, the Screen button will become unfilled and the Printer button will be filled.)

If Screen is selected as the target device, the user must then select the size of the image that will appear on the screen. "Specified Size" can be selected for an image of a certain height and width in pixels, or "Actual Size" for an image on the screen that is the same size as the scanned image. For the "Specified Size" option, only the desired image dimensions (in pixels) need to be entered. For "Actual Size," the user must type the current screen resolution (in pixels) and the Height and Width of the screen measured in inches. (This determines the Screen's dots-per-inch resolution.)

If Printer is selected as the target device, the user must enter the printer's Printer Halftone Resolution, and the final size of the image on the paper (in inches). The printer's Halftone Resolution can usually be found in the printer's

documentation. The final size of the image can usually be determined from the desktop publishing application, or just by measuring a box on the page the same desired size as the scanned image. Since there isn't a direct, one-to-one relationship between pixels and printer halftones, this equation will be an estimation of the best suitable resolution for the printer. Experimenting with the halftone resolution value may yield better results.

Examples:

If a scanned image is to be used in a multimedia presentation, and must be no larger than 600 x 500 pixels, the user would select "Screen," "Specified Size," and enter 600 in the "Image Width" box and 500 in the "Image Height" box.

If a scanned image is to be displayed on a screen as the actual size of the picture to compare video resolutions, the user would select "Screen," "Actual Size," enter the current resolution of the screen (such as 800x600), and enter the physical dimensions of the viewing screen (such as 10.9 inches wide and 8.00 inches high).

If a scanned image is to be used in a newsletter publication in a box that is 3 inches wide and 3 inches deep, the user would select "Printer," enter the halftone resolution (such as 133 lpi), and enter 3 in the Print Width and Print Height boxes.

If, when entering values in either of these dialog boxes, any values are found to be out of range (such as "0"), a message box will inform the user which value is invalid. The user will not be able to save the values to memory if any of them are invalid.

When the values are successfully stored in memory, the user can choose to edit them again by selecting the same dialog box from the menu bar. The stored values will appear in the editing fields, and the user may change them as required.

After all the values have been entered, checked for validity, and stored in memory, the user can now calculate the resolution for the entered data. The "Calculate" option from the menu bar will do this. When this function is executed, the program checks to make sure all of the values have been entered correctly to prevent "divide by zero" errors. Then, it plugs the values into the appropriate equation and displays the result. Following are the equations used to calculate the appropriate scan resolution according to the selected target device:

Screen/Specified Size

$$\frac{\text{image width}}{\text{scanned image width}} = \text{scan resolution 1} \quad \frac{\text{image height}}{\text{scanned image height}} = \text{scan resolution 2}$$

(The value displayed is the lower of the two scan resolutions.)

Screen/Actual Size

$$\frac{\text{greatest dimension of screen resolution}}{\text{screen width (in inches)}} = \text{scan resolution}$$

Printer

$$\frac{\text{printed image height}}{\text{scan image height}} = \text{value 1} \quad \frac{\text{printed image width}}{\text{scan image width}} = \text{value 2}$$

(whichever value is smaller) x halftone frequency x 2 = scan resolution

The value displayed to the user is the vertical and horizontal dots-per-inch resolution that the image should be scanned at. If the target device is either a specified-size image on the screen or a printed image, the picture will be fit to scale entirely within the specified region. For example, if there is a very long, narrow newspaper strip that will be put on a screen image measuring 1024x768 pixels, the image will not be exactly 1024x768 pixels. It may be 1024 pixels wide or 768 pixels high, but the other dimension will always be to scale with the width (or height) of the actual image.

Finally, the "About" option on the menu bar will display the author and version information for the program. From the About dialog box, the user may also view the values in memory. This was included in the program for debugging purposes.

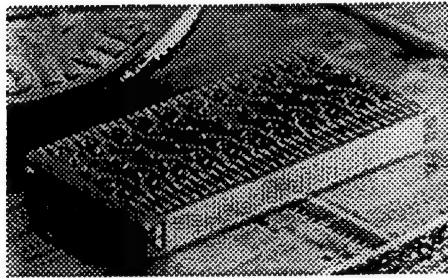
Conclusion

The multimedia advances will change the way people live their lives. Communicating video, information, and learning between locations around the world will be a reality. The potential for sharing of knowledge will increase, and education will undoubtedly benefit. Computers will have the largest role that they've ever had in making the world a better place.

However, before this can be achieved on a large scale, many complex and multifarious problems need to be solved. This work period addressed only a few aspects of new technology individually. Multimedia systems will need many of these components to work together. This will cause a great deal of hardware and software conflicts that will impede the implementation of this fascinating new technology. Until these issues can be resolved and everything can work together in unison, the Information Superhighway will still be under construction.

**DEVELOPING A SOFTWARE
ENVIRONMENT FOR A
HIGH PERFORMANCE
SIGNAL PROCESSOR**

Eric J. Hayduk and Richard A. Schneible Jr.



**Final report for
High School Apprenticeship Program
Rome Laboratory**

**Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.**

and

Rome Laboratory

August 1994

DEVELOPING A SOFTWARE
ENVIRONMENT FOR A
HIGH PERFORMANCE
SIGNAL PROCESSOR

Eric J. Hayduk and Richard A. Schneible Jr.

Abstract

A software environment for the FPASP5 (Floating Point Application Specific Processor v 5.0), a high performance signal processor, was developed. As an integral part of this development, the existing software was evaluated. Much of the Microcode and Assembly language had already been written. Still, debugging was required for all levels of the software environment. Further, function libraries had to be written for use by high level language programmers. Debugging consisted of writing test programs in FPASP5 assembly language, assembling them and simulating the programs on a VHSIC Hardware Description Language (VHDL) model of the FPASP5 processor. Results demonstrated that many tested instructions behaved properly and that several others contained bugs which needed to be documented and corrected. When an instruction worked correctly, in many cases, a function library had to be written. Tested instructions and constructed functions are discussed along with the necessary skills to implement this project, problems, corrections, and future steps to complete the development.

DEVELOPING A SOFTWARE ENVIRONMENT FOR A HIGH PERFORMANCE SIGNAL PROCESSOR

Eric J. Hayduk and Richard A. Schneible Jr.

Introduction

The Wafer Scale Signal Processor (WSSP) and FPASP5 software environment are being developed by the Air Force Rome Laboratory. In recent years, a need has grown for computer processors that have the ability to handle floating point calculations quickly and efficiently. At the same time, it is essential that these processors have small size, low weight, and low power consumption in order to be useful in advanced signal processing applications where these limitations exist.

The WSSP is an adaptable and highly efficient processor that meets these requirements. The fields of signal processing, which involves the detection and tracking of both ground and airborne targets, and supercomputing will benefit greatly from this innovation in technology. This innovation is a result of chip stacking, a streamlined architecture design and hybrid wafer scale integration. Chip stacking involves thinning down the backs of several chips and stacking them on top of one another to save space. Hybrid wafer scale integration is the process of producing chips and then placing them on a wafer, edge to edge, with minimal spacing. The improvements in memory packaging allow for a greater reduction in the size of the WSSP and better memory organization.

The FPASP5 software environment incorporates RISC-like instructions including loads, stores, and simple arithmetic operations. RISC (Reduced Instruction Set Computer) architecture utilizes simpler instructions to increase performance. This occurs because compilers, which optimize programming code, almost always use simpler assembly language instructions rather than more complex assembly language instructions. The FPASP5 software environment also incorporates critical high performance vector routines. The unique FPASP5 architecture and these highly optimized routines allow the WSSP processor to handle vector operations more quickly and efficiently than other signal processors. Function libraries had to be written around these vector commands for use

by high level language programmers.

In the process of completing this evaluation, several skills had to be learned. One of the authors, Eric J. Hayduk, had an intermediate level understanding of the high level programming language C and exposure to assembly language at the start of his tour, while the other author, Richard A. Schneible Jr., had neither. Since these were necessary tools, two weeks were spent learning C. This programming skill was important to learn because it provided a base of programming skills. After two weeks, Richard gained a basic understanding of the C language and Eric had polished his skills. Then it was decided that it was time to learn assembly language. It took only a couple of days for the authors to gain a general understanding of this language, but both learned more with each day of their tours. Another skill that was imperative to have was the ability to use the VI text editor in the UNIX environment. Eric had previously gained experience with the GNU EMACS text editor, but it was important for him to also learn the VI text editor because it was more widely available. After a week of constant usage, the authors had an operational understanding of this editor and its environment. Next, the authors had to learn binary, octal, and hexadecimal notation in order to understand the output which came in these forms. Within a few days both Eric and Richard had gained the necessary experience to work with these notations. Finally, it was necessary to learn to work with commercial products such as WordPerfect, MicroSoft Excel, and PowerPoint. The authors learned how to effectively use these products while constructively utilizing them to document the evaluation of the WSSP software environment.

Methodology and Procedures

By nature, the development of a software environment is a multi-step process. Much of this process had already been completed. Past steps have included designing an accurate simulation of the WSSP processor, writing the Microcode which supports the assembly language instructions, and writing an assembler which has the ability to convert assembly language programs into machine language. When writing any sort of program, however, one must remember that 40 to 60% of the time will be spent debugging.

The process of debugging had not been completed. This consisted of writing test programs, assembling them, simulating them, and reviewing the output to see if the requested operation was correctly performed. The test process began with simple arithmetic computations such as integer and floating point addition, multiplication, and

division. For each operation, values were loaded into several of the registers and output was directed into others. The program was then assembled, that is translated into machine language¹. If a problem arose at this point, it usually meant that either the command was not yet implemented or the parameters were being passed incorrectly. If, however, the program assembled correctly, it was then simulated. The simulator, a VHDL model of the FPASP5 processor, wrote output files that contained data for each clock cycle that was simulated. This data consisted of register usage, the operation codes for the Microcode instruction executed during each clock cycle, and the status of memory at each available address at the end of the simulation. If the output data matched projections, the output was documented and the operation command was added to the list of working commands. On the other hand, if the data did not match projections, other output such as memory dumps and car files were used to pinpoint the bug. As soon as the bug was pinpointed and documented, the information was referred to whoever was in charge of writing that particular piece of code.

The true advantage of the FPASP5 processor is in its ability to handle vector problems rapidly and with accuracy. These operations required more complex tests as there was a larger possibility of bugs. An example of the tested vector operations was CDOT_SP. This command multiplied the individual complex elements of two vectors and then added the results together, that is calculated the complex dot product of two input vectors. In order to effectively test this command, vectors had to be designed with positive, negative, large, and small numbers. This program, once written, was assembled and simulated. When the simulator was finished, the output register was checked for the projected answer. In this case, it was found. So, we concluded that the command did indeed work. In another case, however, HDOT_SP (Hermitian dot product routine) was subjected to a similar test and failed. Its output exactly matched the output from CDOT_SP. HDOT_SP differs from CDOT_SP in that it takes the complex conjugate of each complex element in the first input vector before calculating the dot product. HDOT_SP does use the same Microcode as CDOT_SP. From this, we concluded that a branch condition in the Microcode was not set correctly. This bug was corrected, and the program retested before it was added to the list of working commands. This procedure was followed for all instructions evaluated.

¹ Computers only have the ability to execute these machine language instructions. Even a lower level assembly language such as the FPASP5 must be translated into machine language in order to execute.

A second and equally important part of the development of the software environment was the construction of function libraries in order to facilitate high level language programming. Some of the libraries had already been written in the old assembly language. For these, it was simply necessary to translate the code to be consistent with a new assembly language format. Many of the required changes were uncomplicated such as changing the comment mark from ';' to '#'. Others were more difficult because certain declarations were necessary that weren't before, and other declarations became unnecessary. For example, under the old assembler, symbols were defined using the *.data* statement. Under the new assembler, *.data* was replaced by *.int*, *.float*, and *.double*. These statements were similar in structure and function to the old *.data* statement. Finally, bugs in the new language had to be pinpointed and documented.

The process of debugging these preexisting libraries was essential to the completion of this project. Debugging libraries was very similar to debugging assembly language test routines. The first step was to examine the libraries for any errors in translation. For example, in the new assembly language format, a *.global* statement needed to be added to make subroutines visible to the assembler. The command takes the name of the subroutine as an argument and not the name of the file in which the subroutine is contained. Other simpler errors also occurred involving the incorrect passing of parameters. Errors of these types had to be examined and corrected.

Once a function library assembled correctly, it was simulated on a VHDL model of the WSSP processor and the simulation output was examined. Most of the existing libraries contained bugs which had to be located and corrected. For example, the library CAXPY had to be examined and corrected. It correctly assembled, which means that all of the assembly language commands contained in the library are called correctly. However, when it was simulated, it appeared to go into an endless loop. To confirm this hypothesis, the A_CAR.DAT file was examined. Table 1 is an edited version of the CAR file. Map numbers on the table refer to instructions that are being executed at that particular nanosecond. The instructions stop executing and the simulation continues to run on into infinity. This problem pointed to bugs in the Microcode instructions which support the assembly language library. This information was turned over to the Microcode programmers.

Table 1: Edited CAR file for CAXPY library

```

TIME  MAP
78.5 ns
.
  (Initialization)
.
603.5 ns  1
628.5 ns
653.5 ns  16
678.5 ns
703.5 ns  16
728.5 ns
753.5 ns  16
778.5 ns
803.5 ns  16
828.5 ns
853.5 ns  16
878.5 ns
903.5 ns  16
928.5 ns
953.5 ns  16
978.5 ns
1003.5 ns  14
1028.5 ns
1053.5 ns  8
1078.5 ns
1103.5 ns
1128.5 ns  68
1153.5 ns
1178.5 ns  1
1203.5 ns
1228.5 ns
1253.5 ns  45
1278.5 ns
1303.5 ns
1328.5 ns
1353.5 ns
1378.5 ns
1403.5 ns
1428.5 ns
1453.5 ns
1478.5 ns
1503.5 ns
1528.5 ns
1553.5 ns
1578.5 ns
.
  (to infinity)
.

```

Some of the library function calls had not yet been written even in the old assembly language. For these, it was first necessary to test the main assembly language command around which the library was built. Once that had been proven to work correctly, work on the library function began. The first step was to determine the algorithm involved. The next step was to set up the control of the program, that is the jumps to and from the subroutines. Then, registers had to be set aside to be filled with parameters passed to it by the programmer. Several of the simpler library function required only that a single assembly language command be placed in the primary subroutine. Others required that certain conditions be tested for and then dealt with separately. After the library function was completed, a test shell was written around the function. This program was assembled and simulated in order to test the function. If it worked as expected, it was documented and an entry was added to the manual of library functions.

One of the most difficult libraries that had to be written was Single Precision Vector Plus Vector (SVVP), which added together two vectors composed of real single precision numbers. This was difficult because there were four different possibilities; each had to be dealt with by a separate subroutine. The simplest case occurred when the vector had an even number of elements and was aligned on an eight byte margin. In this case, all that was required was the calling of the assembly language VADD command and the returning, by means of a jump, to the main program. When this case did not occur, at least one element had to be added manually. Even aligned vectors required that the last element in each vector be added manually; odd unaligned, the first; and odd aligned, the first and last. Manually addition was performed by loading the elements to be added from their respective memory addresses into registers and using the FPS_A command to add them. Then, the answer was stored in the appropriate memory address. After this library was completed, a entry was made in a manual of library functions. Table 2 is a flow chart showing the way that SVVP would handle each case. Table 3 shows a graphical representation of the four types of vectors handled by SVVP.

Table 2: SVVP Algorithm Chart

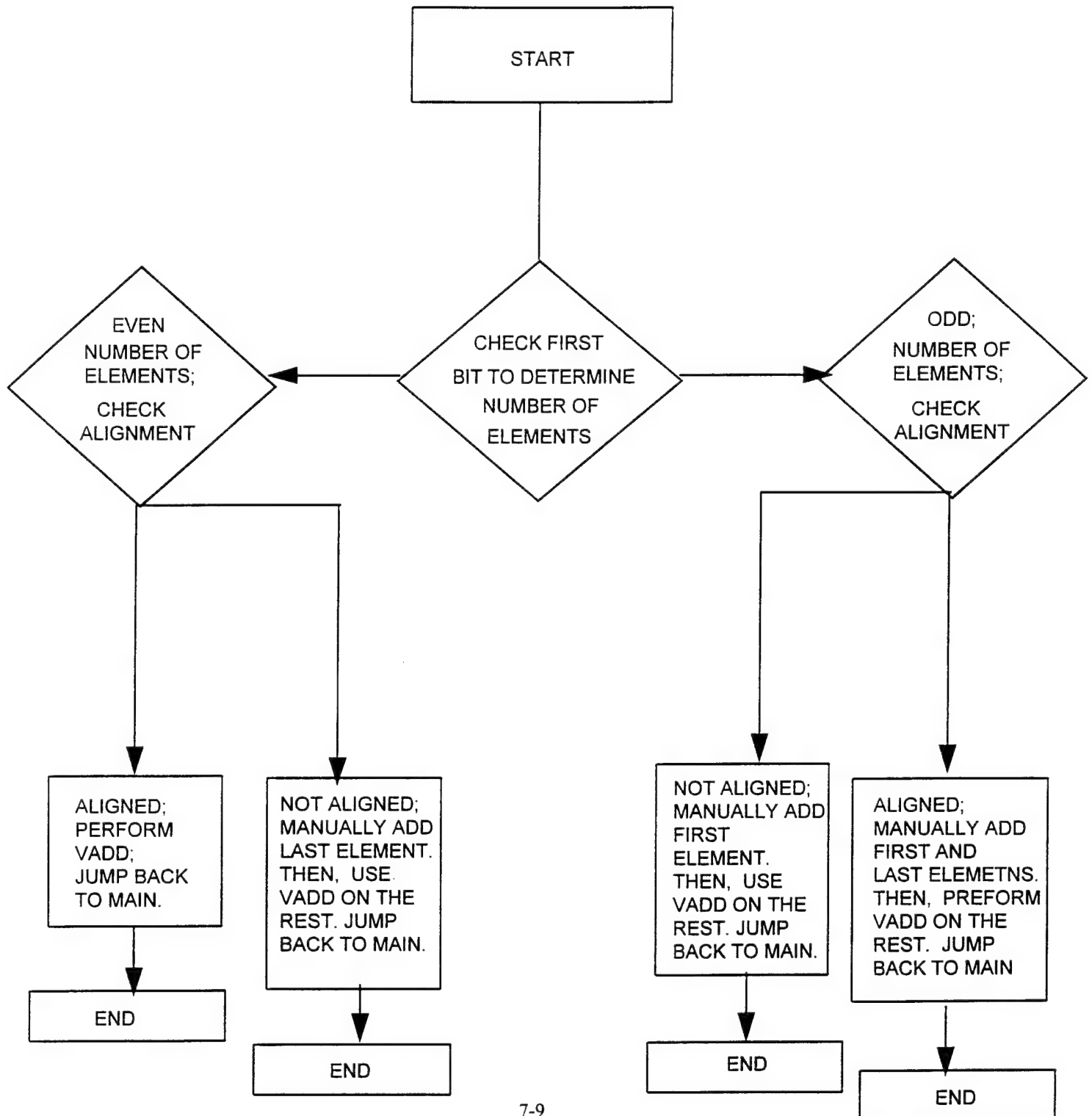


Table 3: Possible input vectors for SVVP

From one side to the other in these boxes represents 8 bytes. Thus, each little box is 4 bytes or the size of a memory address.

1.0	2.0
3.0	4.0
5.0	6.0
7.0	8.0

This is an even and aligned vector.

1.0	2.0
3.0	4.0
5.0	6.0
7.0	

This is an odd and aligned vector.

	1.0
2.0	3.0
4.0	5.0
6.0	7.0
8.0	

This is an even and unaligned vector.

	1.0
2.0	3.0
4.0	5.0
6.0	7.0

This is an odd and unaligned vector.

Results

As previously stated, many of the tested functions worked correctly. These statements did so because there were no errors in the assembly language test routine or the supporting Microcode. Conversely, statements which did not assemble or simulate correctly pointed to errors in the test routine and/or the supporting Microcode. Table 4 summarizes the results of the assembly language command tests. Table 5 summarizes the results of library routines worked on by the authors.

Table 4: Results of Assembly Language Command Tests

Ucode reference	Command Tested	Comments
1. FPADDI	FPS_AI	Assembled and simulated properly on new assembler only
	FPD_AI	Assembled and simulated properly
2. FPMULT	FPS_M	Assembled and simulated properly
	FPS_MM	Assembled and simulated properly
	FPD_M	Assembled and simulated properly
3. FPMI	FPD_MI	Assembled and simulated properly
4. FP_MAC	FPS_MAC	Assembled and simulated properly
	FPS_DMAC	Assembled and simulated properly
	FPD_MAC	BROKE: did not simulate properly due to improperly set FP* bit
5. MULTI	MULTI	Assembled and simulated properly
6. FPADDI2	FPS_DAI	Assembled and simulated properly
7. FPMI2	FPS_DMI	Assembled and simulated properly
8. MULT	MULT	Assembled and simulated properly
9. BR_R	BR_R	Assembled and simulated properly
10. JAL	JAL	Assembled and simulated properly
11. FPMI_SP	FPS_MI	Assembled and simulated properly
12. SRL_LA	SLL	Assembled and simulated properly
	SRL	Assembled and simulated properly
	SRA	Assembled and simulated properly
13. SH	SH	Assembled and simulated properly
	SH (register relative)	Assembled and simulated properly for upper registers only; BROKE: when a store is called with a lower register
14. S_DPT	S_DPT	Assembles and simulates properly

Table 4: Results of Assembly Language Command Tests

15. SETFPBIT	SETFPBIT	Assembled and simulated properly
16. SBLI	SLLI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	SRLI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	ROLI	Assembled and simulated properly except when 0 or any multiple of 32 places were rotated. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	RORI	Assembled and simulated properly except when 0 or any multiple of 32 places were rotated. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
17. SRAI	SRAI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
18. MULTD	MULTD	Assembled and simulated properly
19. MULTDI	MULTDI	BROKE: did not simulate properly
20. CMULT_SP	CMULT_SP	Assembled and simulated properly
21. CDOT_SP	CDOT_SP	Assembled and simulated properly
	HDOT_SP	Assembled and simulated properly
22. CDOT_DP	CDOT_DP	BROKE: assembled but did not simulate properly
23. CVSCALE_SP	CVSCALE	BROKE: assembles but does not simulate properly
24. SP2DP	SP2DP	Assembled and simulated properly
25. written in assembly language	DP2INT	Assembled and simulated properly
26. INT2DP	INT2DP	BROKE: assembled but did not simulate properly
27. MAG_SPV_RM	MAG_SPV_RM	BROKE: assembles but simulates as an endless loop

Table 4: Results of Assembly Language Command Tests

28.VADDSUB	VADD	Assembled and simulated properly
29.VV_MUL	VV_MUL	Assembled and simulated properly
30.MCPY	MCPY	BROKE: assembles but does not simulate properly
31.RDOT_SP	RDOT_SP	BROKE: assembled but simulated as an endless loop
32.RDOT_SP_RM	RDOT_SP_RM	BROKE: assembled but incorrectly simulates for some test cases. It appears that a tie to the lower registers is missing.
33.SETFPR	SETFPR	Assembled and simulated properly
34.SAVEFPR	SAVEFPR	Assembled and simulated properly

Table 5: Status of FPASP5 Libraries as of August 1994

Sub-Routine Name	Library	Ucode Reference	Comments
1. CDOTC_C	BLAS	HDOT_SP	assembled and simulated properly
2. CDOTU_PR	BLAS	CDOT_SP	assembled and simulated properly
3. SETFPMODE	STANDARD	SETFP	assembled and simulated properly
4. CVVP	VECTOR	VADD	assembled and simulated properly
5. CVVM	VECTOR	VSUB	assembled and simulated properly
6. DVVP	VECTOR	VADD	assembled and simulated properly
7. ZVVP	VECTOR	VADD	assembled and simulated properly
8. DVVM	VECTOR	VSUB	assembled and simulated properly
9. ZVVM	VECTOR	VSUB	assembled and simulated properly
10. SVVP	VECTOR	VADD	assembled and simulated properly
11. SVVM	VECTOR	VSUB	assembled and simulated properly

Conclusion

WSSP is a highly adaptable and efficient processor. The future of the WSSP project is dependent upon the completion of this development. After numerous tests and corrections, most of the assembly language commands are now without bugs. The process of debugging should be completed. With this accomplished, the task of writing standard libraries should also be rapidly completed in order that higher level language programmers may work on writing applications. This needs to be done quickly so that WSSP can take advantage of its excellent abilities in the shortest amount of time.

Multi-Media- Creation and Uses
(Using the MacroMind Director
and the NCSA Mosaic)

Michael J. Panara

Rome Free Academy
500 Turin St.
Rome, NY 13440

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1994

Multi-Media- Creation and Uses
(Using the MacroMind Director
and the NCSA Mosaic)

Michael J. Panara
Rome Free Academy

Abstract

Multi-media and it's importance was studied. To do this the MacroMind Director and the NCSA Mosaic applications were used. The Director was used to create a production, and the Mosaic was used to investigate the different ways in which multi-media could be used effectively to get information across to the user. (Mosaic is a tool that is used to let the user "travel" on the World Wide Web.)

Multi-Media-Creation and Uses
(Using the MacroMind Director
and the NCSA Mosaic)

Michael J. Panara

Introduction:

In recent years the use of multi-media has increased dramatically. It has become one of the most popular ways to convey information to the user. It's use of sound, graphics, and animation make the production more interesting and easier for the audience to understand. The number of applications for creating multi-media productions has grown rapidly in the last few years. The one that was used for my presentation was the Director application by MacroMind. Although the application is not the newest on the market, it is probably the most effective and easiest to use.

To see the different ways in which multi-media could be used to get information to the user, the NCSA Mosaic was used. This is an application which allows the user to "travel" throughout the World Wide Web. It allows the user to gather information from various sources throughout the entire world.

Mosaic is the easiest way to travel through the World Wide Web (WWW) because of it's use of hyperlinks. These are highlighted words which when clicked on take the user to another location or document. It also allows the user to move back through pages that have already been accessed. Finally, the Mosaic also has a hot-list that allows the user to save the location of a particular page. This saves the user time next time one chooses to try to find

the same information again.

To better learn about multi-media productions a presentation was created using the MacroMind Director application. The production showed the basic steps involved in making a production using this application. So as the production went along more and more things were learned about the application. The production became more and more involved until eventually the end of the presentation was a fully inter-active multi-media production.

The presentation discussed the basics of the multi-media production, or the essential elements to any multi-media production. These would be text, sound , graphics, animation, and interactivity. It showed how these different elements could be created as well as how these elements are added into the production.

It also discussed the basics of creating the multi-media production, using the Director application. This included instruction on the use of the options of the Director including the cast, score, stage, paint and import options. These options are used when putting together the basic elements into a multi-media production.

All of the elements were created using the paint option or imported from another source. (All sounds had to be imported, since the Director is not capable of creating a sound.) The element was then placed into the cast. Each element becomes a separate cast member and is given a different cast number.

From here the cast member is placed onto the stage. To do this it must be dragged from the cast window onto the stage using the mouse. Once the cast member is placed onto the stage it is automatically placed into the score. This keeps track of all of the cast members and their positions on the stage.

The score also has separate rows to keep other information. This includes the rate at which the production will be played. Also different color palettes can be chosen using the score. All sound is placed in a separate row, unlike the rest of the other cast members.

The most important of these special command rows is the script. This allows the user to type in simple commands for the production. For example the color of the back-ground can be changed. The script also allows the user to delay the production for a any amount of time and then start the production back up again.

Interactivity is also added using the script. First a button is created using the button tool. There are three types of buttons for the user to choose from. After the button is created it is then added to the cast like any other cast member. Then it is dragged onto the stage. Without adding the proper script the button would only be just another graphic. When the proper commands are added to the script the button then allows the viewer to decide which path the production will take.

Sometimes the user may want to connect two related movies. Instead of having to copy all of one movie and then transfer into the other movie, the movies can be connected using the script. When the first movie ends, the Director will automatically load the next movie. There is no limit to the amount of movies that can be connected.

These are just the basic steps in creating a multi-media presentation. There are also ways to make a transition from one screen smoothly by fading one out and the next screen in. This and other little things can be done to make the production look more professional.

The NCSA Mosaic makes great use of multi-media. All pages on the Mosaic are done in multi-media. They contain text, graphics, sound, and interactivity; some even contain animation.

The Mosaic itself does not actually use multi-media. The Mosaic is just a tool that allows the user navigate through the World Wide Web. The pages that the Mosaic allows the user to connect to are the elements that actually utilize multi-media.

The Mosaic is an easy way to access information from around the world. Any server that is connected to the World Wide Web anywhere in the world can be accessed using the NCSA Mosaic.

It is a completely interactive network. The Mosaic utilizes links known as hyperlinks. All the user has to do to change pages is to click the mouse on one of the hi-lighted phrases. This is known as hyper-text. Sometimes a graphic is surrounded by a border and the border is hi-lighted; this is also a hyperlink.

The Mosaic can be used to access libraries from around the world. It also can access various colleges and universities. This is especially useful for doing reports or research, or to look through university catalogs to help with making the decision about which college to attend.

Another feature that makes the Mosaic easy to use is it's ability to navigate backward through documents that have already been accessed. Also once the user goes back one can then go forward to the farthest point accessed.

The uses of multi-media are ever increasing, as one can see. The Mosaic is a fairly new application that makes full use of this new technology. It shows how useful multi-media can be in presenting information, and how much easier it is to understand information presented in this manner.

The Director shows how a fully interactive and very effective multi-media presentation can be created with relative ease. One can make their presentation much more interesting to the viewer by using this application.

Multi-media will continue to grow in the future. It's many uses make it a valuable tool for anyone who is presenting a report or any other type of information.

STUDY OF GLOBAL
HYPERMEDIA NETWORKS

Anne E. Pletl
Engineering Assistant For Software
Exercises And Verification
Griffiss Air Force Base (Rome Labs)

Rome Laboratory
Communications Network Branch
525 Brooks Road
GAFB NY 13441-4505

Final Report For:
Summer Research Program
Rome Laboratory

Sponsored By:
Air Force Office Of Scientific Research
Bolling Air Force Base

August 1994

**STUDY OF GLOBAL
HYPERMEDIA NETWORKS**

**Anne E. Pletl
Rome Laboratory
Communications Networks Branch
Griffiss Air Force Base**

Abstract

Global hypermedia networks which access hypermedia servers (i.e. text, audio, and video) were studied and demonstrated during my apprenticeship this summer at Rome Labs. In my research I have found that through the use of these revolutionary new projects, which are continually being upgraded, it is possible to contact another database (server) anywhere else in the world (given the proper address and hookup). Specifically, I developed a hypermedia server describing work performed at the Rome Lab Network Design Facility (NDF). I also updated several files, which enable the NDF easier access to the available information on the network. The following report will discuss my involvement with these hypermedia networks, the hypermedia server running on a SUN workstation, and explain their capabilities now, as well as their potential for the future.

A STUDY OF GLOBAL HYPERMEDIA NETWORKS

Anne E. Pletl

INTRODUCTION

Global hypermedia networks address the needs of government, industry, and even the public, by providing a more accessible way to get information. To understand how they operate you must explore the meaning of hypermedia or hypertext that includes or links to other forms of media. A basic understanding of the project known as the World Wide Web, the Internet which it all runs on, and the html language is also needed. The content of this report further explains the global hypermedia program and the work that is being done with it.

THE INTERNET

The Internet is made up of thousands of smaller regional networks scattered throughout the globe. It is one massive world-wide network of computers. The program most frequently used on the Internet is the World-Wide Web due to its accessibility and wide range of functions. It is important not

to confuse the two however. The Internet refers to the tangible side of the global network, like the cables, switches, routers, etc. Whereas the Web is more abstract, like the information itself.

WORLD-WIDE WEB

The official description of the Web is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents , according to Kevin Hughes of Honolulu Community College. Its discovery, in March of 1989 was by Tim Berners-Lee of CERN, the European Laboratory for particle physics. His main goal was to provide the research community with an effective transportation system for ideas to each other around the world (since CERN's members are located in a number of countries). This project has provided users on computer networks with a consistent means to access a variety of media in a simplified fashion.

The Web runs under a client-server model (a client interfaces with the user to request documents from the server). Thousands of virtual transactions take place every hour throughout the world.

Months after the Web's invention, the National Center for Supercomputing Applications (NCSA) began a project to create an interface in the World-Wide Web. Its original concern was to

help the scientific research community by producing widely available, non-commercial software. What they created, a global hypermedia network, exceeded all expectations. In 1993, a mouse driven interface called Mosaic was introduced to the Internet community, already with a small yet strong following.

MOSAIC

Mosaic is an Internet-based global hypermedia browser that allows you to discover, retrieve, and display documents and data from all over the Internet. Put simply, this is a program that allows the user to interact with information, by accessing any file in the Internet. Figure 1 shows the Mosaic "home page" I developed for the NDF. The underlined text, which is displayed in color on a computer screen, is actually hypertext. When the user clicks on this hypertext, additional information about that subject is provided. Thus a user can quickly "browse" through information. The hypertext can also be information in the form of pictures, video, and audio clips.

Figure 1.



C3

Network Design Facility

C3

• The Network Design Facility performs research, development, and testing of network architectures, protocols, application, and management.

Resources include:

- Asynchronous Transfer Mode (ATM) testbed
which incorporates the Fore Systems and GTE ATM switches, ROMENET, a packet switched network composed of BBN C-30 and C-3 packet switches, and COCONET, an internetwork of various switching technologies. These networking capabilities are integrated into an internetworked environment with access to the Defense Simulation Internet.
- Joint Advanced Development Environment (JADE)
- NYNet
an ATM based network developed in cooperation with industry and academia. The facility also includes the Network Performance Assessment Environment which provides network management capabilities for all of the resources, and an Error Injector Unit which allows emulation of a channel's error characteristics, and allows for testing of network topologies and protocols over disadvantaged links.
- ATM Hardware
- GTE SPANet ATM Hardware

THE HYPERTEXT MARKUP LANGUAGE (HTML)

The standard language the Web uses for creating and recognizing hypermedia documents, like Mosaic, is the hypertext markup language (html) (Figure 2).

Figure 2

```
<TITLE>NDF</TITLE>
<PRE>
<IMG SRC="http://166.101.10.104/c3-home.gif">
<P>
<IMG SRC="http://166.101.10.104/c3-bar.gif">
<H2>Network Design Facility</H2>
<IMG SRC="http://166.101.10.104/c3-bar.gif"><BR>
</PRE>
<P>
<IMG SRC="http://166.101.10.104/greenball.gif">
The Network Design Facility
  performs research, development, and testing of
  network architectures, protocols, application, and management. Resources
  include:
<UL>
<IMG SRC="http://166.101.10.104/greenball.gif"><A HREF="http://166.101.10.104/
Asynchronous Transfer Mode (ATM) testbed</A>
<UL>which incorporates
the Fore Systems and GTE ATM switches, ROMENET, a packet switched network
composed of BBN C-30 and C-3 packet switches, and COCONET, an internetwork
of various switching technologies. These networking capabilities are
integrated into an internetworked environment with access to the Defense
Simulation Internet.
</UL>
<P>
<IMG SRC="http://166.101.10.104/greenball.gif"><A HREF="http://166.101.10.104/
Joint Advanced Development Environment (JADE)</A>
<P>
<IMG SRC="http://166.101.10.104/greenball.gif"><A HREF="http://166.101.10.104/
NYNet</A>
<UL>
an ATM based network developed in cooperation with industry and
academia. The facility also includes the Network Performance Assessment
Environment which provides network management capabilities for all of the
resources, and an Error Injector Unit which allows emulation of a
channel's error characteristics, and allows for testing of network
topologies and protocols over disadvantaged links.</UL></UL>
<P>
<IMG SRC="http://166.101.10.104/greenball.gif"><A HREF="http://166.101.10.104/
ATM Hardware</A>
<P>
<IMG SRC="http://166.101.10.104/greenball.gif"><A HREF="http://166.101.10.104/
GTE SPANet ATM Hardware</A>
```

It uses Uniform Resource Locators (URLs) to represent almost every file connected to the network (Figure 3).

Figure 3

http://166.101.10.104/ATMTestbed.html

The first part of the URL http, (HyperText Transmission Protocol), is the standard language that the World-Wide Web clients and servers use to communicate. The next part is the address and the last part specifies the file at that address that is trying to be accessed (usually ends with the suffix .html).

COMPLICATIONS

As with all computer programs, several problems arose. The main problem was trying to incorporate pictures into html language and the Mosaic program. The program would not accept any pictures unless they were in the proper format. In this case they were gifs which is a Graphic Interchange Format. Figure 4 shows html code which points to the Command, Control, and Communicate Directorate gif shown in figure 1.

Figure 4

Another problem that arose with the pictures was that In order to insert them into the Mosaic program, they all had to be .gif files. The problem was that all the pictures available were not in that format (e.g. tiff, xbm, wpg). I alleviated that problem by creating a .gif file and saved the other format into that file. It was then a .gif file and could be run on Mosaic.

The final complication I encountered was the actual text. That too had to be in the correct format (Figure 5). One type would create a problem and not allow the program to be accessed in Mosaic.

Figure 5

```
<A HREF="http://166.101.10.104/ATMTestbed.html">  
Asynchronous Transfer Mode (ATM) Testbed</A>
```

(In the figure above the Asynchronous Transfer Mode (ATM) Testbed would be what was seen on the Mosaic screen.)

CONCLUSION

There are many benefits to using global hypermedia networks. It allows the user to contact any file within the Internet. With the new technology out today, these hypermedia projects are the wave of the future, part of the information super highway.

I envision in a few years we will have the capabilities to

contact anywhere else in the world for any kind of information. Classes will be able to be conducted in foreign languages, and across oceans. Businesses will be able to contact their foreign counterparts to improve enterprise. Governments will be able to communicate with each other without having to hassle with complex forms. And scientific research will be available to anyone with the know-how to obtain it. I feel that within the next few years, libraries will be obsolete because the information will be at everyone's fingertips.

Great strides are being taken every day due to the use of the hypermedia project. The research community can only benefit from this global hypermedia project and the public will in turn be affected. Maybe disease will be decreased, or famine. Once the knowledge of this project is more publicized, I feel we can begin to challenge ourselves in ways not thought possible ten years ago.

WORKS CITED

Hughes, Kevin. Entertaining The World-Wid Web: A Guide To Cyberspace. Honolulu Community College. (c)October 1993.

Itano, Wayne M. Getting Started On Mosaic . Optics and Photonics News (c)June 1994 p48.

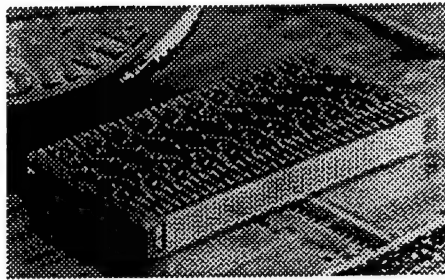
McCarthy, Shawn P. What Direction Will Mosaic Take? Government Computer News (c)June 13, 1994 p80.

McCarthy, Shawn P. MacWeb challenges Mosaic As Freeware For Internet". Government Computer News (c)July 25, 1994 p61.

Vaughan-Nichols, Steven J. How To Glue Together Mosaic . Government Computer News : Technology Report (c)July 18, 1994 pp. 33, 36-37.

**DEVELOPING A SOFTWARE
ENVIRONMENT FOR A
HIGH PERFORMANCE
SIGNAL PROCESSOR**

Eric J. Hayduk and Richard A. Schneible Jr.



**Final report for
High School Apprenticeship Program
Rome Laboratory**

**Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.**

and

Rome Laboratory

August 1994

DEVELOPING A SOFTWARE
ENVIRONMENT FOR A
HIGH PERFORMANCE
SIGNAL PROCESSOR

Eric J. Hayduk and Richard A. Schneible Jr.

Abstract

A software environment for the FPASP5 (Floating Point Application Specific Processor v 5.0), a high performance signal processor, was developed. As an integral part of this development, the existing software was evaluated. Much of the Microcode and Assembly language had already been written. Still, debugging was required for all levels of the software environment. Further, function libraries had to be written for use by high level language programmers. Debugging consisted of writing test programs in FPASP5 assembly language, assembling them and simulating the programs on a VHSIC Hardware Description Language (VHDL) model of the FPASP5 processor. Results demonstrated that many tested instructions behaved properly and that several others contained bugs which needed to be documented and corrected. When an instruction worked correctly, in many cases, a function library had to be written. Tested instructions and constructed functions are discussed along with the necessary skills to implement this project, problems, corrections, and future steps to complete the development.

DEVELOPING A SOFTWARE
ENVIRONMENT FOR A
HIGH PERFORMANCE
SIGNAL PROCESSOR

Eric J. Hayduk and Richard A. Schneible Jr.

Introduction

The Wafer Scale Signal Processor (WSSP) and FPASP5 software environment are being developed by the Air Force Rome Laboratory. In recent years, a need has grown for computer processors that have the ability to handle floating point calculations quickly and efficiently. At the same time, it is essential that these processors have small size, low weight, and low power consumption in order to be useful in advanced signal processing applications where these limitations exist.

The WSSP is an adaptable and highly efficient processor that meets these requirements. The fields of signal processing, which involves the detection and tracking of both ground and airborne targets, and supercomputing will benefit greatly from this innovation in technology. This innovation is a result of chip stacking, a streamlined architecture design and hybrid wafer scale integration. Chip stacking involves thinning down the backs of several chips and stacking them on top of one another to save space. Hybrid wafer scale integration is the process of producing chips and then placing them on a wafer, edge to edge, with minimal spacing. The improvements in memory packaging allow for a greater reduction in the size of the WSSP and better memory organization.

The FPASP5 software environment incorporates RISC-like instructions including loads, stores, and simple arithmetic operations. RISC (Reduced Instruction Set Computer) architecture utilizes simpler instructions to increase performance. This occurs because compilers, which optimize programming code, almost always use simpler assembly language instructions rather than more complex assembly language instructions. The FPASP5 software environment also incorporates critical high performance vector routines. The unique FPASP5 architecture and these highly optimized routines allow the WSSP processor to handle vector operations more quickly and efficiently than other signal processors. Function libraries had to be written around these vector commands for use

by high level language programmers.

In the process of completing this evaluation, several skills had to be learned. One of the authors, Eric J. Hayduk, had an intermediate level understanding of the high level programming language C and exposure to assembly language at the start of his tour, while the other author, Richard A. Schneible Jr., had neither. Since these were necessary tools, two weeks were spent learning C. This programming skill was important to learn because it provided a base of programming skills. After two weeks, Richard gained a basic understanding of the C language and Eric had polished his skills. Then it was decided that it was time to learn assembly language. It took only a couple of days for the authors to gain a general understanding of this language, but both learned more with each day of their tours. Another skill that was imperative to have was the ability to use the VI text editor in the UNIX environment. Eric had previously gained experience with the GNU EMACS text editor, but it was important for him to also learn the VI text editor because it was more widely available. After a week of constant usage, the authors had an operational understanding of this editor and its environment. Next, the authors had to learn binary, octal, and hexadecimal notation in order to understand the output which came in these forms. Within a few days both Eric and Richard had gained the necessary experience to work with these notations. Finally, it was necessary to learn to work with commercial products such as WordPerfect, MicroSoft Excel, and PowerPoint. The authors learned how to effectively use these products while constructively utilizing them to document the evaluation of the WSSP software environment.

Methodology and Procedures

By nature, the development of a software environment is a multi-step process. Much of this process had already been completed. Past steps have included designing an accurate simulation of the WSSP processor, writing the Microcode which supports the assembly language instructions, and writing an assembler which has the ability to convert assembly language programs into machine language. When writing any sort of program, however, one must remember that 40 to 60% of the time will be spent debugging.

The process of debugging had not been completed. This consisted of writing test programs, assembling them, simulating them, and reviewing the output to see if the requested operation was correctly performed. The test process began with simple arithmetic computations such as integer and floating point addition, multiplication, and

division. For each operation, values were loaded into several of the registers and output was directed into others. The program was then assembled, that is translated into machine language¹. If a problem arose at this point, it usually meant that either the command was not yet implemented or the parameters were being passed incorrectly. If, however, the program assembled correctly, it was then simulated. The simulator, a VHDL model of the FPASP5 processor, wrote output files that contained data for each clock cycle that was simulated. This data consisted of register usage, the operation codes for the Microcode instruction executed during each clock cycle, and the status of memory at each available address at the end of the simulation. If the output data matched projections, the output was documented and the operation command was added to the list of working commands. On the other hand, if the data did not match projections, other output such as memory dumps and car files were used to pinpoint the bug. As soon as the bug was pinpointed and documented, the information was referred to whoever was in charge of writing that particular piece of code.

The true advantage of the FPASP5 processor is in its ability to handle vector problems rapidly and with accuracy. These operations required more complex tests as there was a larger possibility of bugs. An example of the tested vector operations was CDOT_SP. This command multiplied the individual complex elements of two vectors and then added the results together, that is calculated the complex dot product of two input vectors. In order to effectively test this command, vectors had to be designed with positive, negative, large, and small numbers. This program, once written, was assembled and simulated. When the simulator was finished, the output register was checked for the projected answer. In this case, it was found. So, we concluded that the command did indeed work. In another case, however, HDOT_SP (Hermitian dot product routine) was subjected to a similar test and failed. Its output exactly matched the output from CDOT_SP. HDOT_SP differs from CDOT_SP in that it takes the complex conjugate of each complex element in the first input vector before calculating the dot product. HDOT_SP does use the same Microcode as CDOT_SP. From this, we concluded that a branch condition in the Microcode was not set correctly. This bug was corrected, and the program retested before it was added to the list of working commands. This procedure was followed for all instructions evaluated.

¹ Computers only have the ability to execute these machine language instructions. Even a lower level assembly language such as the FPASP5 must be translated into machine language in order to execute.

A second and equally important part of the development of the software environment was the construction of function libraries in order to facilitate high level language programming. Some of the libraries had already been written in the old assembly language. For these, it was simply necessary to translate the code to be consistent with a new assembly language format. Many of the required changes were uncomplicated such as changing the comment mark from ';' to '#'. Others were more difficult because certain declarations were necessary that weren't before, and other declarations became unnecessary. For example, under the old assembler, symbols were defined using the *.data* statement. Under the new assembler, *.data* was replaced by *.int*, *.float*, and *.double*. These statements were similar in structure and function to the old *.data* statement. Finally, bugs in the new language had to be pinpointed and documented.

The process of debugging these preexisting libraries was essential to the completion of this project. Debugging libraries was very similar to debugging assembly language test routines. The first step was to examine the libraries for any errors in translation. For example, in the new assembly language format, a *.global* statement needed to be added to make subroutines visible to the assembler. The command takes the name of the subroutine as an argument and not the name of the file in which the subroutine is contained. Other simpler errors also occurred involving the incorrect passing of parameters. Errors of these types had to be examined and corrected.

Once a function library assembled correctly, it was simulated on a VHDL model of the WSSP processor and the simulation output was examined. Most of the existing libraries contained bugs which had to be located and corrected. For example, the library CAXPY had to be examined and corrected. It correctly assembled, which means that all of the assembly language commands contained in the library are called correctly. However, when it was simulated, it appeared to go into an endless loop. To confirm this hypothesis, the A_CAR.DAT file was examined. Table 1 is an edited version of the CAR file. Map numbers on the table refer to instructions that are being executed at that particular nanosecond. The instructions stop executing and the simulation continues to run on into infinity. This problem pointed to bugs in the Microcode instructions which support the assembly language library. This information was turned over to the Microcode programmers.

Table 1: Edited CAR file for CAXPY library

```

TIME  MAP
78.5 ns
.
  (Initialization)
.
603.5 ns  1
628.5 ns
653.5 ns  16
678.5 ns
703.5 ns  16
728.5 ns
753.5 ns  16
778.5 ns
803.5 ns  16
828.5 ns
853.5 ns  16
878.5 ns
903.5 ns  16
928.5 ns
953.5 ns  16
978.5 ns
1003.5 ns  14
1028.5 ns
1053.5 ns  8
1078.5 ns
1103.5 ns
1128.5 ns  68
1153.5 ns
1178.5 ns  1
1203.5 ns
1228.5 ns
1253.5 ns  45
1278.5 ns
1303.5 ns
1328.5 ns
1353.5 ns
1378.5 ns
1403.5 ns
1428.5 ns
1453.5 ns
1478.5 ns
1503.5 ns
1528.5 ns
1553.5 ns
1578.5 ns
.
  (to infinity)
.

```

Some of the library function calls had not yet been written even in the old assembly language. For these, it was first necessary to test the main assembly language command around which the library was built. Once that had been proven to work correctly, work on the library function began. The first step was to determine the algorithm involved. The next step was to set up the control of the program, that is the jumps to and from the subroutines. Then, registers had to be set aside to be filled with parameters passed to it by the programmer. Several of the simpler library function required only that a single assembly language command be placed in the primary subroutine. Others required that certain conditions be tested for and then dealt with separately. After the library function was completed, a test shell was written around the function. This program was assembled and simulated in order to test the function. If it worked as expected, it was documented and an entry was added to the manual of library functions.

One of the most difficult libraries that had to be written was Single Precision Vector Plus Vector (SVVP), which added together two vectors composed of real single precision numbers. This was difficult because there were four different possibilities; each had to be dealt with by a separate subroutine. The simplest case occurred when the vector had an even number of elements and was aligned on an eight byte margin. In this case, all that was required was the calling of the assembly language VADD command and the returning, by means of a jump, to the main program. When this case did not occur, at least one element had to be added manually. Even aligned vectors required that the last element in each vector be added manually; odd unaligned, the first; and odd aligned, the first and last. Manually addition was performed by loading the elements to be added from their respective memory addresses into registers and using the FPS_A command to add them. Then, the answer was stored in the appropriate memory address. After this library was completed, a entry was made in a manual of library functions. Table 2 is a flow chart showing the way that SVVP would handle each case. Table 3 shows a graphical representation of the four types of vectors handled by SVVP.

Table 2: SVVP Algorithm Chart

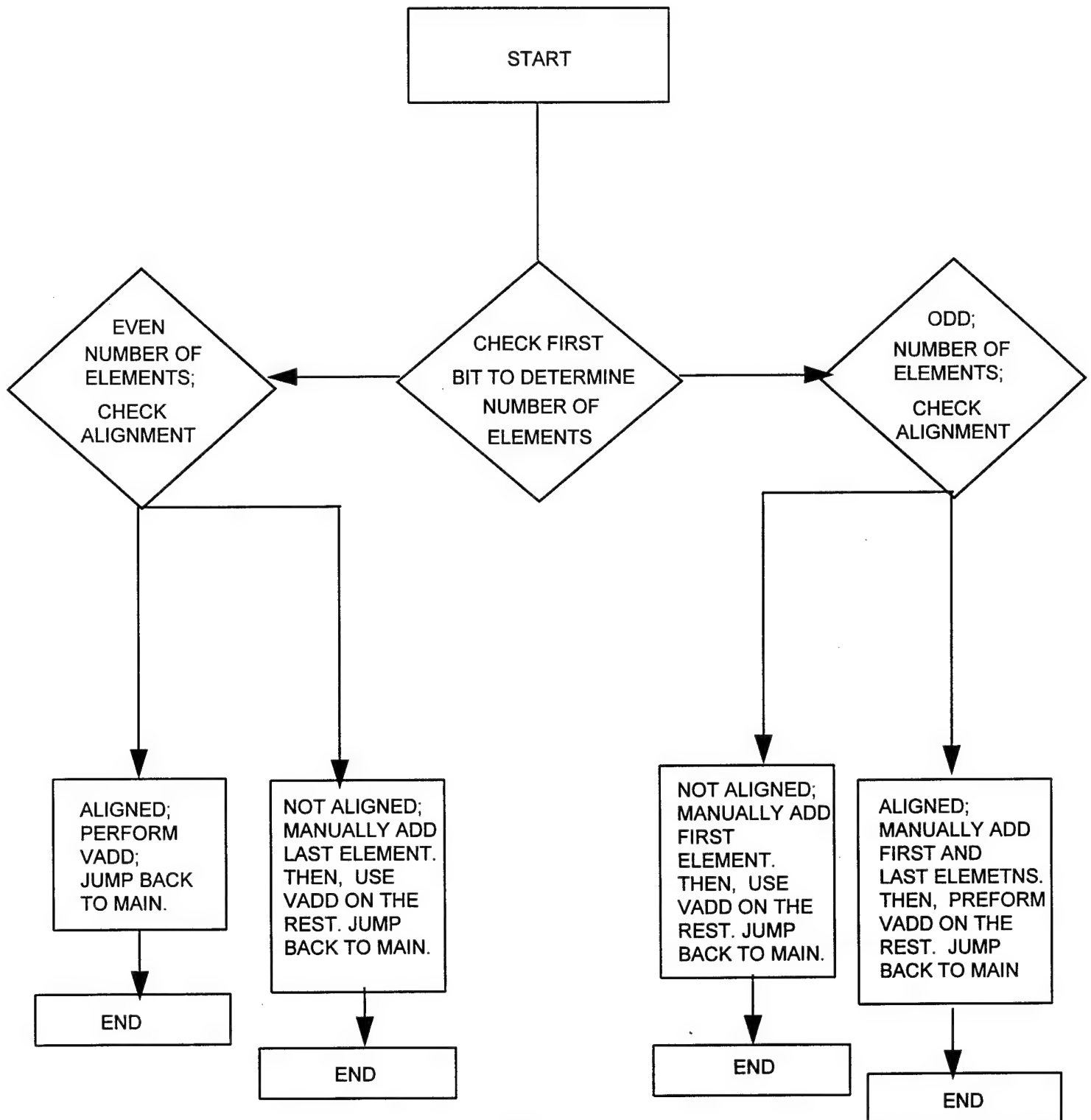


Table 3: Possible input vectors for SVVP

From one side to the other in these boxes represents 8 bytes. Thus, each little box is 4 bytes or the size of a memory address.

1.0	2.0
3.0	4.0
5.0	6.0
7.0	8.0

This is an even and aligned vector.

1.0	2.0
3.0	4.0
5.0	6.0
7.0	

This is an odd and aligned vector.

	1.0
2.0	3.0
4.0	5.0
6.0	7.0
8.0	

This is an even and unaligned vector.

	1.0
2.0	3.0
4.0	5.0
6.0	7.0

This is an odd and unaligned vector.

Results

As previously stated, many of the tested functions worked correctly. These statements did so because there were no errors in the assembly language test routine or the supporting Microcode. Conversely, statements which did not assemble or simulate correctly pointed to errors in the test routine and/or the supporting Microcode. Table 4 summarizes the results of the assembly language command tests. Table 5 summarizes the results of library routines worked on by the authors.

Table 4: Results of Assembly Language Command Tests

Ucode reference	Command Tested	Comments
1. FPADDI	FPS_AI	Assembled and simulated properly on new assembler only
	FPD_AI	Assembled and simulated properly
2. FPMULT	FPS_M	Assembled and simulated properly
	FPS_MM	Assembled and simulated properly
	FPD_M	Assembled and simulated properly
3. FPMI	FPD_MI	Assembled and simulated properly
4. FP_MAC	FPS_MAC	Assembled and simulated properly
	FPS_DMAC	Assembled and simulated properly
	FPD_MAC	BROKE: did not simulate properly due to improperly set FP* bit
5. MULTI	MULTI	Assembled and simulated properly
6. FPADDI2	FPS_DAI	Assembled and simulated properly
7. FPMI2	FPS_DMI	Assembled and simulated properly
8. MULT	MULT	Assembled and simulated properly
9. BR_R	BR_R	Assembled and simulated properly
10. JAL	JAL	Assembled and simulated properly
11. FPMI_SP	FPS_MI	Assembled and simulated properly
12. SRL_LA	SLL	Assembled and simulated properly
	SRL	Assembled and simulated properly
	SRA	Assembled and simulated properly
13. SH	SH	Assembled and simulated properly
	SH (register relative)	Assembled and simulated properly for upper registers only; BROKE: when a store is called with a lower register
14. S_DPT	S_DPT	Assembles and simulates properly

Table 4: Results of Assembly Language Command Tests

15. SETFPBIT	SETFPBIT	Assembled and simulated properly
16. SBLI	SLLI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	SRLI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	ROLI	Assembled and simulated properly except when 0 or any multiple of 32 places were rotated. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
	RORI	Assembled and simulated properly except when 0 or any multiple of 32 places were rotated. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
17. SRAI	SRAI	Assembled and simulated properly except when 0 or any multiple of 32 places were shifted. This occurs because the masking value becomes zero and the barrel shifter won't drive the C bus with a zero shift. Occurs as expected
18. MULTD	MULTD	Assembled and simulated properly
19. MULTDI	MULTDI	BROKE: did not simulate properly
20. CMULT_SP	CMULT_SP	Assembled and simulated properly
21. CDOT_SP	CDOT_SP	Assembled and simulated properly
	HDOT_SP	Assembled and simulated properly
22. CDOT_DP	CDOT_DP	BROKE: assembled but did not simulate properly
23. CVSCALE_SP	CVSCALE	BROKE: assembles but does not simulate properly
24. SP2DP	SP2DP	Assembled and simulated properly
25. written in assembly language	DP2INT	Assembled and simulated properly
26. INT2DP	INT2DP	BROKE: assembled but did not simulate properly
27. MAG_SPV_RM	MAG_SPV_RM	BROKE: assembles but simulates as an endless loop

Table 4: Results of Assembly Language Command Tests

28.VADDSUB	VADD	Assembled and simulated properly
29.VV_MUL	VV_MUL	Assembled and simulated properly
30.MCPY	MCPY	BROKE: assembles but does not simulate properly
31. RDOT_SP	RDOT_SP	BROKE: assembled but simulated as an endless loop
32. RDOT_SP_RM	RDOT_SP_RM	BROKE: assembled but incorrectly simulates for some test cases. It appears that a tie to the lower registers is missing.
33.SETFPR	SETFPR	Assembled and simulated properly
34. SAVEFPR	SAVEFPR	Assembled and simulated properly

Table 5: Status of FPASP5 Libraries as of August 1994

Sub-Routine Name	Library	Ucode Reference	Comments
1. CDOTC_C	BLAS	HDOT_SP	assembled and simulated properly
2. CDOTU_PR	BLAS	CDOT_SP	assembled and simulated properly
3. SETFPMODE	STANDARD	SETFP	assembled and simulated properly
4. CVVP	VECTOR	VADD	assembled and simulated properly
5. CVVM	VECTOR	VSUB	assembled and simulated properly
6. DVVP	VECTOR	VADD	assembled and simulated properly
7. ZVVP	VECTOR	VADD	assembled and simulated properly
8. DVVM	VECTOR	VSUB	assembled and simulated properly
9. ZVVM	VECTOR	VSUB	assembled and simulated properly
10. SVVP	VECTOR	VADD	assembled and simulated properly
11. SVVM	VECTOR	VSUB	assembled and simulated properly

Conclusion

WSSP is a highly adaptable and efficient processor. The future of the WSSP project is dependent upon the completion of this development. After numerous tests and corrections, most of the assembly language commands are now without bugs. The process of debugging should be completed. With this accomplished, the task of writing standard libraries should also be rapidly completed in order that higher level language programmers may work on writing applications. This needs to be done quickly so that WSSP can take advantage of its excellent abilities in the shortest amount of time.

ADESH as a Sample Generator for mdem

Nathan B. Terry

Clinton Sr. High School
Chenengo Ave.
Clinton, N.Y. 13323

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air force Base, DC

and

Rome Laboratory

August 1994

ADESH as a Sample Generator for mdem

Nathan B. Terry
Clinton Sr. High School

Abstract

ADESH (Atomistic DEfect Simulation Handler), developed by CASA (Center for Simulations and Analysis), has been found here to be a very useful tool for providing the mdem (molecular dynamics electromigration) simulator with polycrystalline samples. The main reason for ADESH's success at providing mdem with samples is its versatility. Such versatility allows a variety of polycrystalline cells to be created for the mdem system. Creating the methodology to use ADESH as a sample generator for mdem is very critical as mdem cannot generate complex samples. Thus ADESH will prove to be an integral element for the mdem system.

ADESH as a Sample Generator for mdem

Nathan B. Terry

Introduction

When sufficient electric current is passed through an aluminum interconnect, hillocks and voids form. This phenomenon, called electromigration, is known to be a contributing factor to the failure of integrated circuits. It is of great interest to investigate this problem in the hope that some way can be found to minimize its effects.

There are several ways to study electromigration. One way is through the use of microscopies. This can be very time consuming, however, because electromigration occurs at unpredictable locations. An additional problem is that electromigration occurs beneath the surface of a sample, an area microscopies generally have difficulty studying. A second, less difficult way to study electromigration is through the use of computer simulators¹. One such program is the mdem (molecular dynamics electromigration) simulator, created by Dr. Herb Helbig. This program investigates electromigration using Newton's law, $F=ma$, to simulate the molecular dynamics of a crystal. However, the mdem simulator does not create the samples it uses to simulate electromigration. Instead, another simulator program, ADESH (Atomistic DEfect Simulation Handler) produced by CASA (Center for Atomic Simulation and Analysis), was found here to be a very useful tool in performing this task. The program's versatility allows a variety of crystal samples to be produced for the mdem system.

Discussion of ADESH

The first step in the ADESH sample creation process is to design the desired sample which is to be studied. This structure, known in ADESH as the computational cell (CC), can be composed of a maximum of 2000 atoms of any element or combination of elements. The operator can also designate the lattice structure of the unit cell: simple

cubic, body-centered cubic, face-centered cubic, diamond cubic. The user can also design his own crystal lattice, one with up to thirty atom positions per unit cell.

Once the crystal structure is fixed, the user can also determine the orientation of the crystal by entering the orientation in the form of the crystal's Miller indices. Once the Miller indices are entered, the computer creates a perfect crystal of essentially infinite dimensions, one atom of which is located at the coordinate origin. The user can then make the CC by entering its dimensions. Whichever atoms fall inside the dimensions are included in the CC.

The CC can have a variety of shapes, the most basic being box-like, cylindrical and spherical. Through creative manipulations of these simple shapes, others can be created. For example, a wedge-shaped CC can be created by making a cylinder with a limiting angle of less than 2π radians. Likewise, cones can be created by constructing partial spheres. Polycrystalline CCs can be constructed by using combinations of these shapes. Grain boundaries can be created by adjusting the Miller index of each monocrystal to be different from that of its neighbors. Additionally, the ADESH system can create CCs composed of alloys or CCs which have vacancies at random sites. There is also a command which allows individual atoms to be manipulated, so as to tailor the CC to the specific needs of the user.

ADESH stores the CC in the computer as a compilation of three coordinates (X, Y, and Z) for each atom, along with the atomic number to identify the atom type. The program records the location of voids by assigning them an atomic number of zero. The usefulness of such a list of coordinates is its versatility. This allows the CC to be examined in several different ways.

One way to analyze the CC is by using the View command of the ADESH program. The atoms of the CC can be color coded by type or by plane, the latter command giving each atomic layer parallel to the Z plane a different color than its

neighboring planes. Once the operator has selected the color coding of the CC, he can then choose either an XY plot, an XZ plot or a YZ plot. One of the limitations of the ADESH View command is its inability to give a three dimensional plot. Fortunately, however, the list of atomic coordinates the ADESH simulator produces can be read into other plotting programs which will display three dimensional plots. SigmaPlot, a spreadsheet program, was used to plot the bi-crystals in three dimensions.

Another advantage of the list of atomistic coordinates ADESH produces is that these coordinates can also be transferred to other simulators. This enables the program to generate samples for the molecular dynamics simulator used in this project.

Simulation

Two samples were created in an approximate rectangular parallelepiped composed of two pie-shaped wedges (see appendix A, B). Each wedge, an FCC (Face-Centered Cubic) aluminum crystal, had a different crystal orientation. The two wedges of each sample were combined in a box-like arrangement designed to minimize the Lennard-Jones potential energy. The energy was minimized by moving the two wedges relative to each other and computing the sum of the Lennard-Jones potential over all the atom pairs after each translation until a minimum was found.

The two samples were then transferred to the molecular dynamics simulator, mdem. The sample was relaxed via Newton's law, $F=ma$. The simulation heated the sample from a temperature of 0K to 10K with 5000 integration steps of 2 fs per step. There were a total of 1896 atoms in the CC. A second, cumulative relaxation of the sample raised the temperature from 10K to 100K with an additional 5000 integration steps of 2fs per step.

Results

A careful comparison between the bi-crystal produced by ADESH and the same bi-crystals after a relaxation by mdem demonstrates that the CC did relax. In the ADESH

produced bi-crystal, there was a noticeable gap between the two wedges that composed the CC. After the first mdem relaxation, this gap was transformed into a grain boundary where the two wedges merged together (See Appendix C, D). A further mdem relaxation of the ADESH crystal, one in which the temperature reached almost 700K, the two wedges merged even more extensively (See Appendix E, F). (Although the second mdem simulation almost reached a temperature of 700K, the thermostat function of the simulation was set to 100K, which is the reason the sample temperature was reported from 10K to 100K in the preceding paragraph).

Conclusions

The results obtained by the mdem simulator demonstrate that the bi-crystals produced by ADESH are usable by the mdem system. ADESH can, therefore, be used as a sample generator for mdem. Because of ADESH's flexibility, there are many possible polycrystalline samples which can be produced. The ADESH samples can be of any atom type, any crystal lattice, any orientation and any shape which has fewer than 2000 atoms. ADESH will therefore prove to be critical for providing samples for the mdem program.

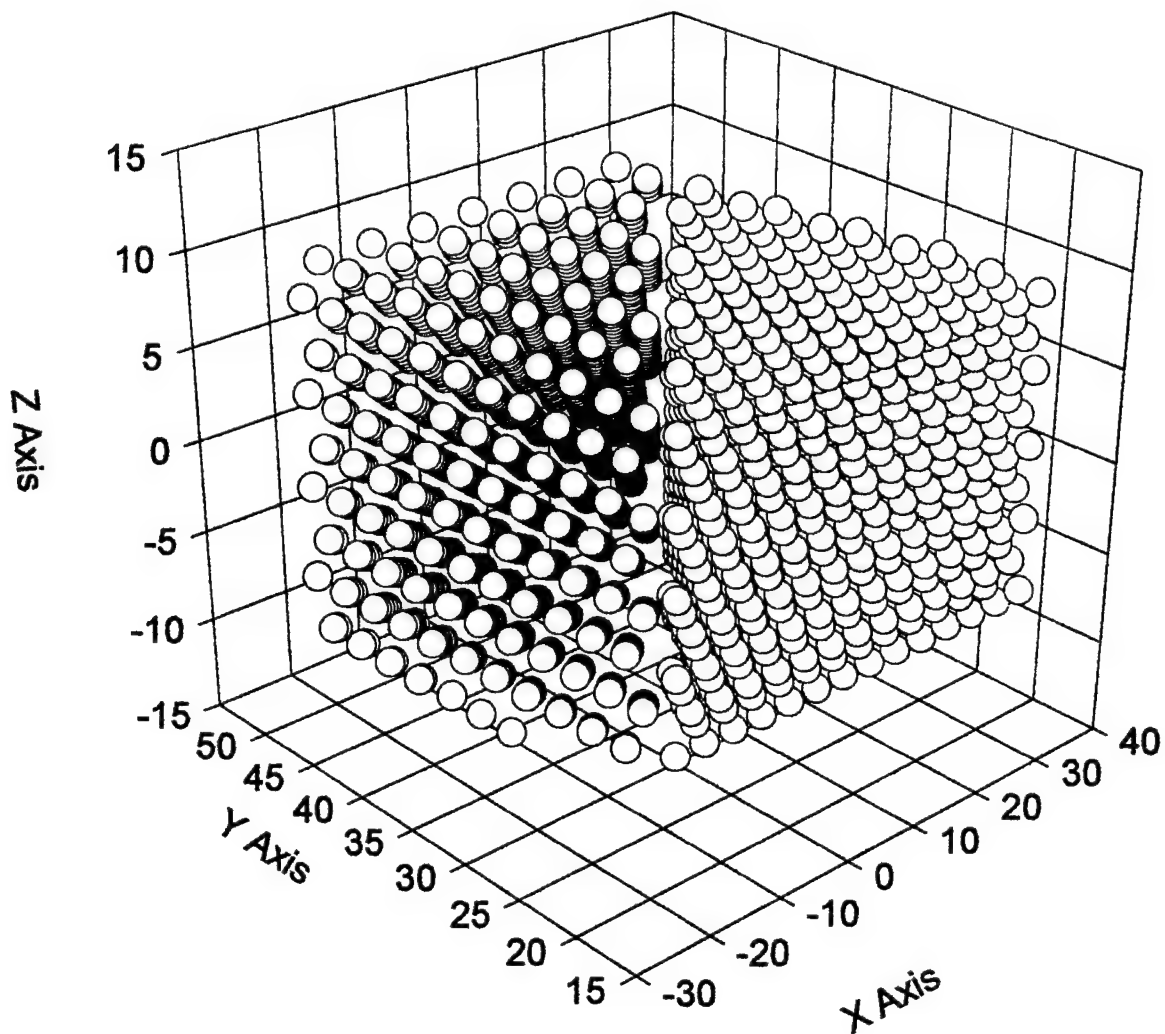
References

1. H.F. Helbig, T. Bartelt, L.H. Walsh, J.V. Beasock, IEEE Dual-Use Conference 1994, 98.

Appendix A

Bi-Crystal produced by ADESH

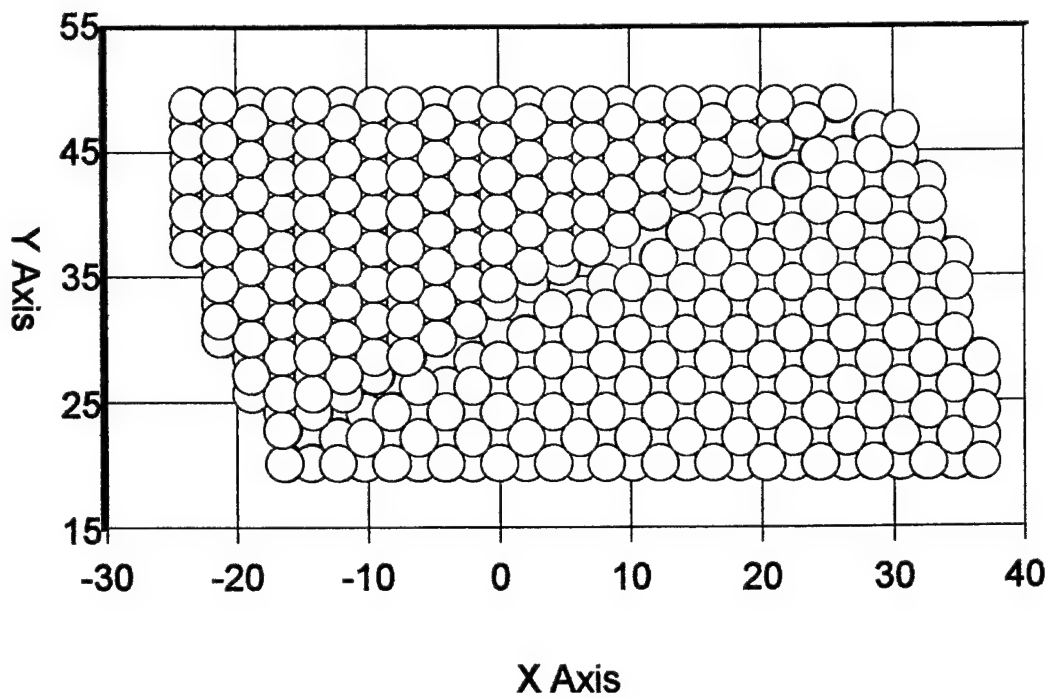
1896 aluminum atoms



Appendix B

Bi-Crystal produced by ADESH

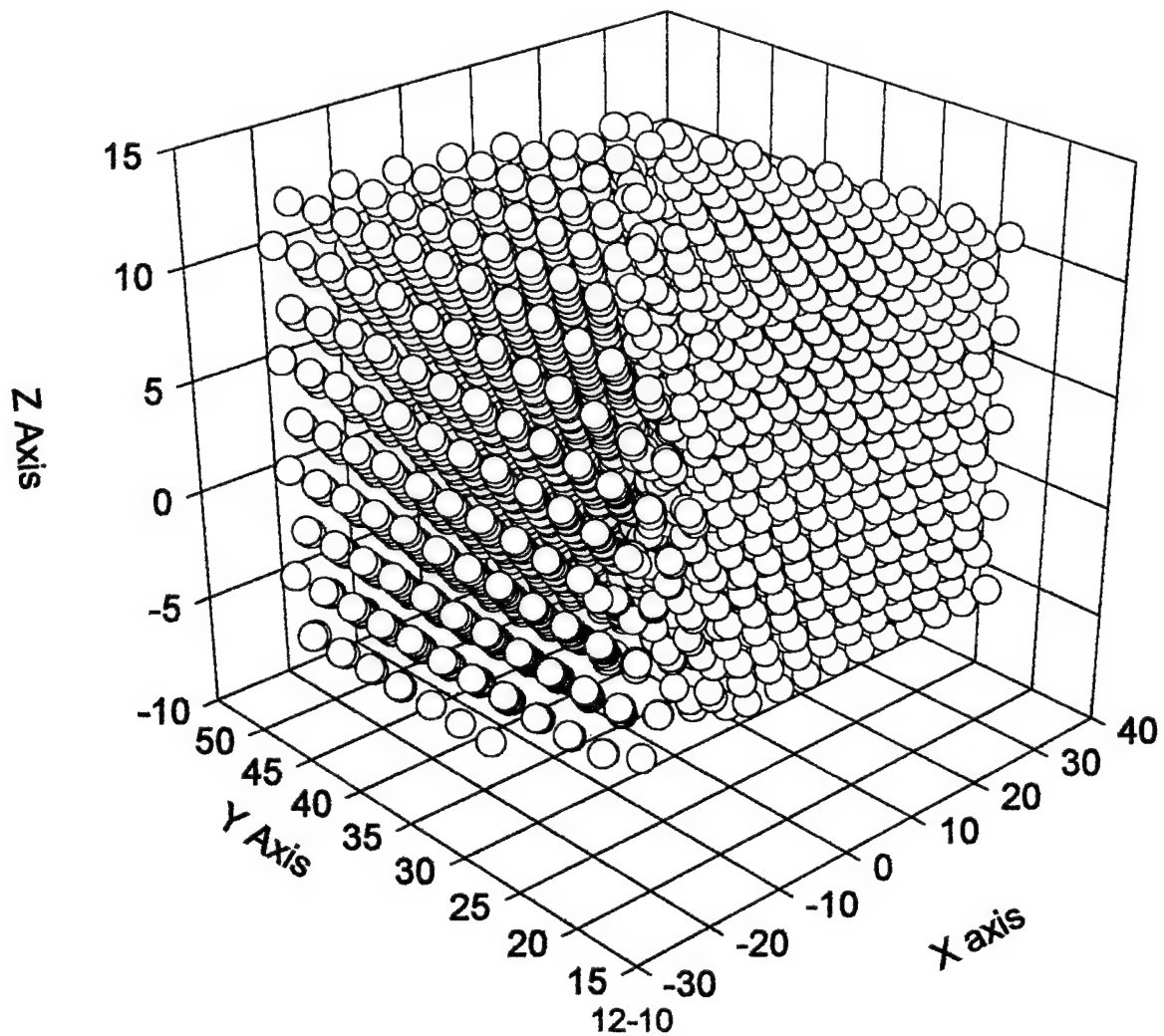
1896 aluminum atoms



Appendix C

ADESH Bi-Crystal after 60 hrs (10ps) in mdem 0K to 10K

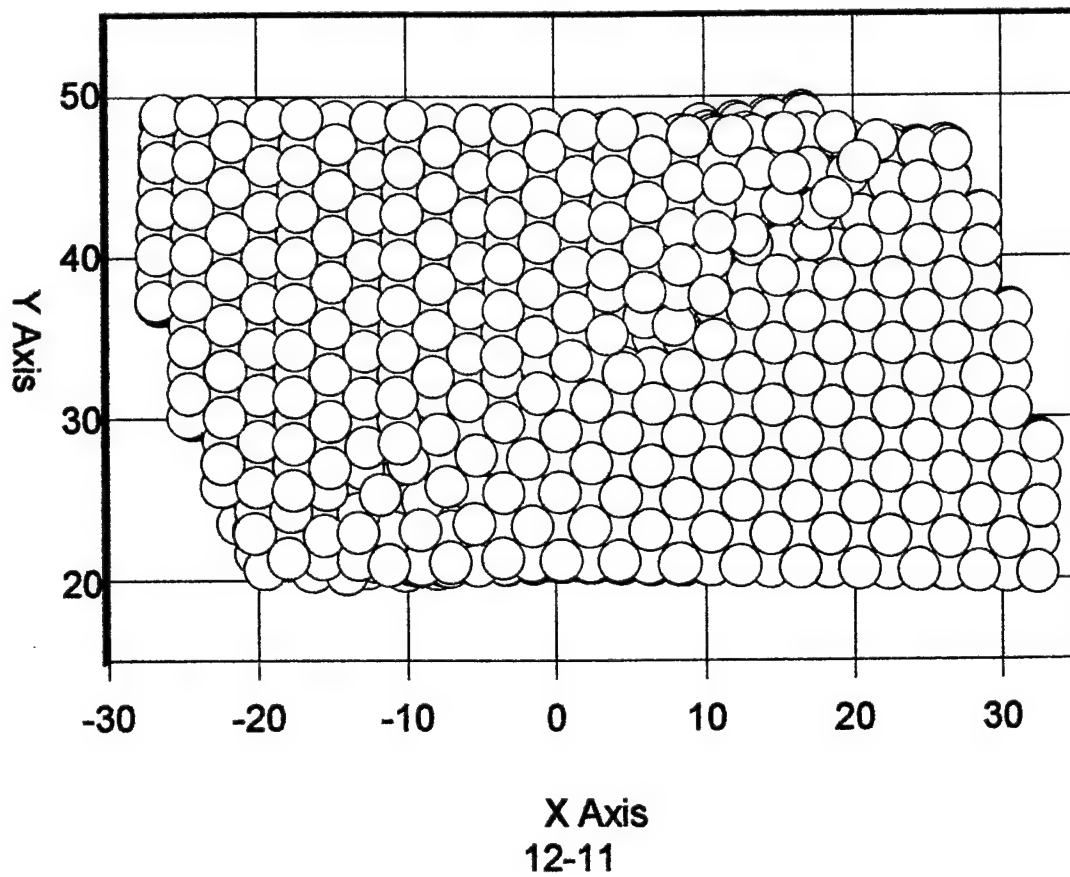
1896 aluminum atoms



Appendix D

ADESH Bi-Crystal after 60 hrs (10ps) in mdem 0K to 10K

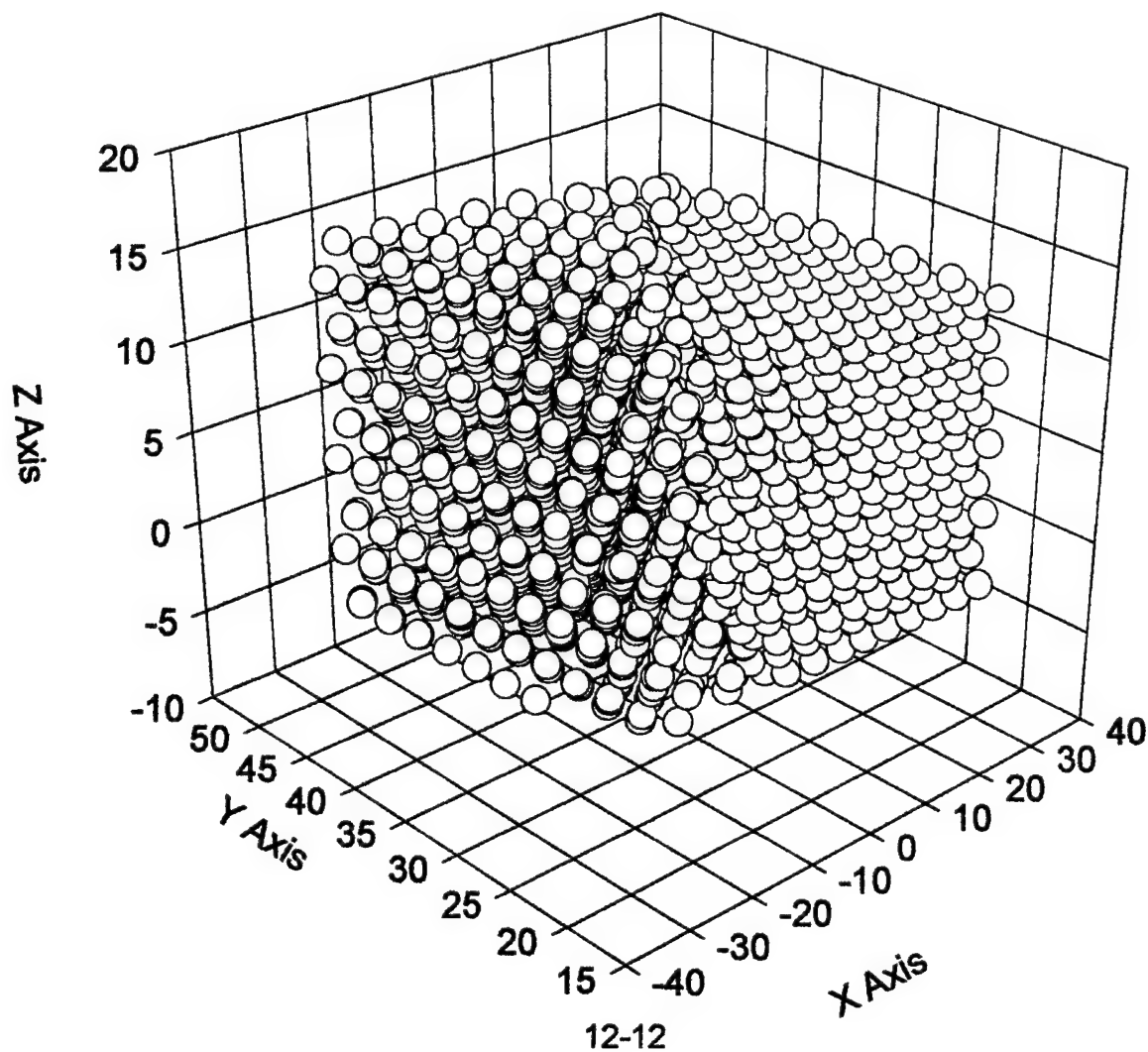
1896 aluminum atoms



Appendix E

ADESH Bi-Crystal after 120hrs (20 ps) in mdem 0K to 100K

1896 aluminum atoms

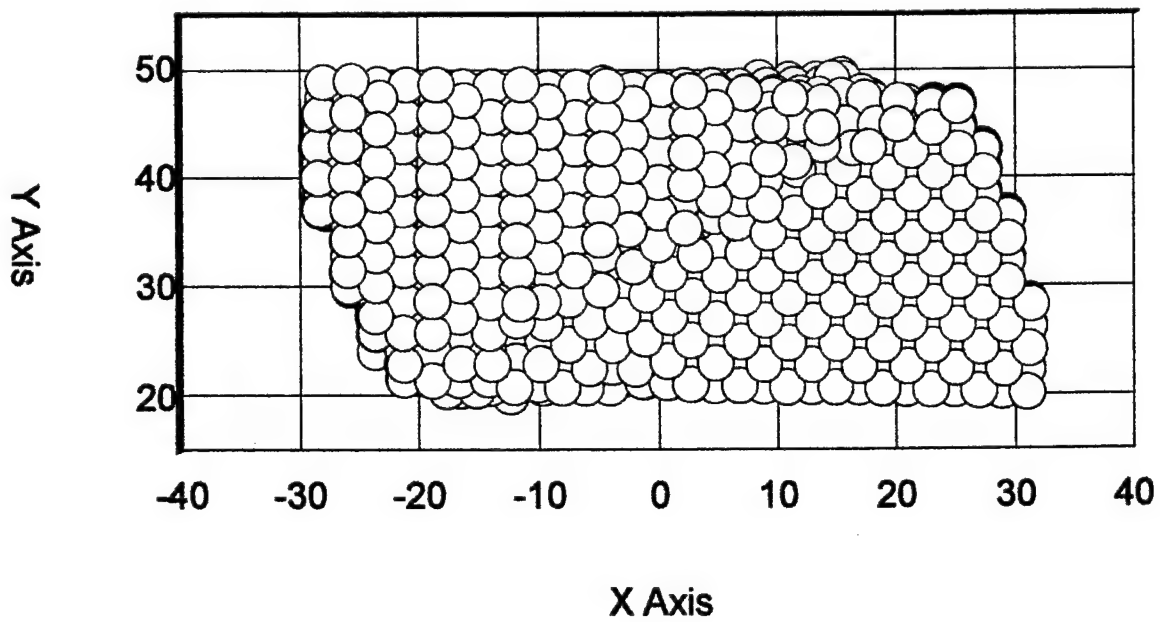


Appendix F

ADESH Bi-Crystal after 120hrs (20 ps) in mdem

0K to 100K

1896 aluminum atoms



The Physical significance of the Eigenvalues in Adaptive Arrays

Brian Testa

New Hartford Senior High
33 Oxford Road
New Hartford, NY 13413

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air force Base, DC

and

Rome Laboratory

August 1994

The Physical Significance of the Eigenvalues in Adaptive Arrays

BRIAN P.V. TESTA ,
Rome Laboratory
VINCENT C. VANNICOLA ,
Rome Laboratory

Abstract

We describe the physical significance of the eigenvalues associated with an adaptive antenna array. We also show how the eigenvalues are affected by the power output of the noise sources, the location of the noise sources, and the spatial configuration of the array antenna. The model used to show these relationships consists of two array elements and two independent noise sources located in the far field. We show how the eigenvalues of the covariance matrix vary with the angle of incidence of the noise sources.

I. Introduction

Adaptive array receiving antennas have been of interest in the fields of communication and radar for several years. In the past [1] [2] and even to this day, the operation of the adaptive array process depends a great deal upon the properties of the noise covariance matrix and its associated eigenvalues. When adaptive processes become large in dimensionality, people may use

various methods to reduce the rank of such matrices [3]. We find large dimensionality in large arrays or when adaptive processing occurs over multiple signal domains, e.g. space time processing [4]. In an adaptive array, a separate coherent output is obtained from each element(or sub-array) of a phased-array receiving antenna. The output of each element channel is sampled and multiplied by a complex weight, i.e., adjusted in both amplitude and phase, so that while the beam is steered in a desired look direction, nulls are placed in the direction of noise sources. The complex weights in the element channels are controlled by time variant algorithms which adapt to the signal/noise environment. This means that the illumination function of the receiving array antenna is controlled adaptively. The nature of the eigenvalues play an important role in how the adaptive process responds. Hudson [5] presented general expressions for eigenvalues of a two element colinear array.

In this paper, we describe in detail the physical significance of the eigenvalues associated with an adaptive array. Using a two element array, we also show how the eigenvalues are affected by varying the power output of noise sources, the location of these noise sources, and the spatial configuration of the array antenna. Section II will review the mathematical description of an eigenvalue and the transformation process represented thereof. In Section III we derive the covariance matrix of a two element antenna array. This will be a function of the parameters governing the array system and the signal noise environment. In Section IV we derive

the eigenvalues of that matrix. Finally, in Section V, we show some results and draw conclusions about how the eigenvalues behave to changes in the array configuration and the noise environment .

II. Eigenvalues

The eigenvalues and covariance matrix of an adaptive array are the central theme in this paper. Therefore, we shall briefly review the concepts of linear algebra [6] that pertain to these topics with some relevance to their physical significance.

An eigenvalue is defined as a scalar c such that:

$$A\alpha = c\alpha \quad (1)$$

where A is an $n \times n$ matrix and $\alpha \neq 0$ a vector. Therefore, c is an eigenvalue of A . In order to illustrate how we arrive at the values of c we provide the following example: $A = \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix}$. In order to find

the eigenvalues of A we must find all scalars c such that:

$$\begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = c \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad (2a)$$

or

$$\begin{aligned} 2a_1 + 2a_2 &= ca_1 \\ 2a_1 + 5a_2 &= ca_2 \end{aligned} \quad (2b)$$

This homogeneous system of two equations and two unknowns has a nontrivial solution if and only if the

determinant of the coefficient matrix is zero. Thus:

$$\begin{vmatrix} c-2 & -2 \\ -2 & c-5 \end{vmatrix} = 0 \quad (3)$$

This means that $c^2 - 7c + 6 = 0$. Solving for c we obtain the eigenvalues of matrix A are $c_1 = 1$ and $c_2 = 6$.

To find some physical significance to what the matrix, A , and its eigenvalues, c , represent in a signal processing situation, let us assume that the above A is the covariance matrix arising from the two zero mean noise signals, y_1 and y_2 (random variables) at the input terminals of a network. This may be represented by the random vector,

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (4)$$

The covariance matrix is the expectation (average) of Y with its Hermetian. i.e.,

$$A = E\{Y Y^H\} = E \left\{ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{bmatrix} y_1^* & y_2^* \end{bmatrix} \right\} \quad (5)$$

where E is the expectation (averaging) operation, H is conjugate transpose where the $*$ denotes the complex conjugate. Carrying out the above operations and assumption

$$A = E \begin{bmatrix} y_1 y_1^* & y_1 y_2^* \\ y_2 y_1^* & y_2 y_2^* \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \quad (6)$$

where the numerical values are the corresponding product averages. A is therefore the covariance matrix of the random vector, Y .

Figure 1 shows the physical system described by the above equations. For an input consisting of the two variables in Y with covariances shown in equation (6), we may obtain an output

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (7)$$

whose covariance matrix, C , is a diagonal matrix, simply by operating on Y by the transformation(modal) matrix, T . Hence, as in (5) and (6)

$$C = E\{\mathbf{X} \mathbf{X}^t\} = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix} \quad (8a)$$

$$T = 1/(5)^{1/2} \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \quad (8b)$$

It can be shown that the transformation

$$\mathbf{X} = T\mathbf{Y} \quad (9)$$

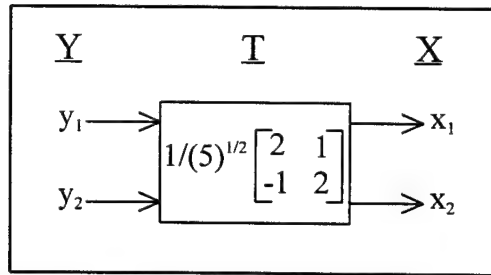


Figure 1: Transformation Block Diagram

transforms the vector Y whose elements y_1, y_2 are correlated with one another to the vector X whose elements are uncorrelated with one another. In terms of signal processing, noise sources y_1, y_2 , having covariance matrix A , were linear combinations of the two independent noise sources denoted by x_1, x_2 and having covariance matrix C . The

transformation, T , has converted the y_1, y_2 back to their original individual independent components, x_1, x_2 . Verification (9) of this result is left to the reader.

In finding T , one must determine the eigenvectors of A . This procedure may be found in the literature [7].

III. Determination of Covariance Matrix for a Two Element Antenna Array

Now that we have reviewed the basic concepts needed to understand this topic, we give the example on which most of this paper will be based. Let us assume that there exist two elements in an array receiving r_1 and r_2 , respectively. In the far field, there also exist two independent complex gaussian zero mean noise sources (n_1 and n_2). *note: It should be made clear, that n_1 and n_2 are completely independent of each other.* The model for this situation is depicted in Figure 2a.

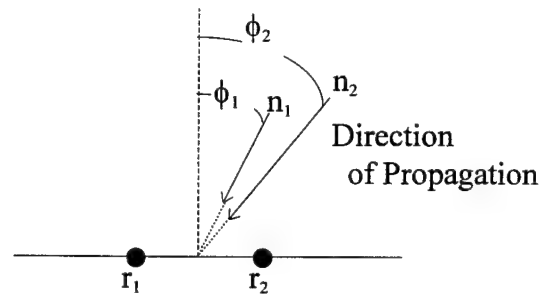


Figure 2a. Two Element Array Model

As it is illustrated in the diagram, ϕ_1 and ϕ_2 are the incident angles of n_1 and n_2 as they are measured from the normal. These angles, along with the

amplitude of the noise sources will directly affect r_1 , r_2 and the eigenvalues of the array covariance matrix.

The covariance matrix is determined by taking the expectation of the outer product of \mathbf{r} and \mathbf{r}^H as was done in (5) produces

$$\mathbf{M} = E\{\mathbf{r} \mathbf{r}^H\} \quad (10)$$

where: $\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$ and $\mathbf{r}^H = [r_1^* \ r_2^*]$

where * denotes the complex conjugate. With the aid of figure 2b we determine the values of r_1 and r_2 . [8]

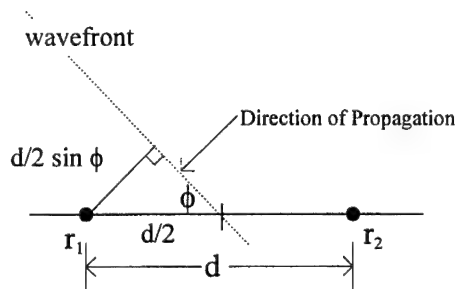


Figure 2b. Wavefront Incident on Array

The spatial delay of the wavefront incident on the array at angle ϕ_1 is $d/2 \sin \phi_1$ for element r_1 . Since the electrical phase per unit length in the direction of wave propagation is $k=2\pi/\lambda$ radians, then the electrical phase delay with respect to the array center k denotes the propagation constant, λ is the wavelength (see **Figure 2b**).

With this information, we can establish the values of r_1 and r_2 as:

$$\begin{aligned} r_1 &= n_1 e^{(-jkd \sin \phi_1)/2} + n_2 e^{(-jkd \sin \phi_2)/2} \\ r_2 &= n_1 e^{(jkd \sin \phi_1)/2} + n_2 e^{(jkd \sin \phi_2)/2} \end{aligned} \quad (11)$$

where $j = (-1)^{1/2}$, $k = 2\pi/\lambda$, d = distance between r_1 and r_2 .

We are now in a position to evaluate the covariance matrix, \mathbf{M} . As in (5)

$$\mathbf{M} = E\{\mathbf{r} \mathbf{r}^H\}. \quad (12)$$

note: The following identities have already been proven for n_1 and n_2 mutually independent variables. These expressions are expanded upon in the text. [9]

$$\begin{aligned} E\{n_i n_j^*\} &= 0 \\ E\{n_i n_i^*\} &= N_i : i = 1, 2 \end{aligned} \quad (13)$$

Now we will determine the covariance matrix for r_1, r_2 . Expanding (12) we have

$$\mathbf{M} = E \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \begin{bmatrix} r_1^* & r_2^* \end{bmatrix} \quad (14)$$

Carrying out the multiplication in (14)

$$\mathbf{M} = E \begin{bmatrix} r_1 r_1^* & r_1 r_2^* \\ r_2 r_1^* & r_2 r_2^* \end{bmatrix} \quad (15)$$

Substituting for r_1 , r_2 , r_1^* , and r_2^* , carrying out the expectation, and simplifying, we get

$$\mathbf{M} = \begin{bmatrix} N_1 + N_2 & N_1 e^{-jkd \sin \phi_1} + N_2 e^{-jkd \sin \phi_2} \\ N_1 e^{jkd \sin \phi_1} + N_2 e^{jkd \sin \phi_2} & N_1 + N_2 \end{bmatrix} \quad (16)$$

N_1 and N_2 are the variances of the zero mean complex gaussian random signals n_1 and n_2 , respectively.

IV. Determination of Array Eigenvalues

Now that we have determined the covariance matrix of the array, we can now evaluate the eigenvalues c_i . In order to accomplish this, we follow the procedure used in the previous section. First, we multiply the matrix M by an eigenvector α and set it equal to c times the same vector.

$$\begin{bmatrix} N_1 + N_2 & N_1 e^{-jkd \sin \phi_1} + N_2 e^{-jkd \sin \phi_2} \\ N_1 e^{jkd \sin \phi_1} + N_2 e^{jkd \sin \phi_2} & N_1 + N_2 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} = c \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} \quad (17a)$$

OR

$$\begin{aligned} (N_1 + N_2)a_{11} + (N_1 e^{-jkd \sin \phi_1} + N_2 e^{-jkd \sin \phi_2})a_{21} &= ca_{11} \\ (N_1 e^{jkd \sin \phi_1} + N_2 e^{jkd \sin \phi_2})a_{11} + (N_1 + N_2)a_{21} &= ca_{21} \end{aligned} \quad (17b)$$

Following the procedures set forth in (2) and (3), we evaluate the determinant

$$\begin{vmatrix} N_1 + N_2 - c & N_1 e^{-jkd \sin \phi_1} + N_2 e^{-jkd \sin \phi_2} \\ N_1 e^{jkd \sin \phi_1} + N_2 e^{jkd \sin \phi_2} & N_1 + N_2 - c \end{vmatrix} = 0 \quad (18)$$

The determinant of this matrix yields the equation

$$c^2 - (2N_1 + 2N_2)c + 2N_1 N_2 - N_1 N_2 (e^{jkd(\sin \phi_1 - \sin \phi)} + e^{jkd(\sin \phi_2 - \sin \phi)}) = 0 \quad (19)$$

Since this equation is in the form $ac^2 + bc + x = 0$, we can evaluate c by using the quadratic formula:

$$c = \frac{-b \pm (b^2 - 4ax)^{1/2}}{2a} \quad (20)$$

$$a = 1$$

$$b = -(2N_1 + 2N_2)$$

$$x = 2N_1 N_2 - N_1 N_2 (e^{jkd(\sin \phi_1 - \sin \phi)} + e^{jkd(\sin \phi_2 - \sin \phi)})$$

Once all substitutions have been made, the unsimplified equation is:

$$c = (N_1 + N_2) \pm [N_1^2 + 2N_1 N_2 + N_2^2 - N_1 N_2 (2 - e^{jkd(\sin \phi_1 - \sin \phi_2)} + e^{-jkd(\sin \phi_1 - \sin \phi_2)})]^{1/2} \quad (21)$$

Through basic simplification, we compress the equation into the following format.

$$c = N_1 + N_2 \pm [N_1^2 + N_2^2 - N_1 N_2 (e^{jkd(\sin \phi_1 - \sin \phi_2)} + e^{-jkd(\sin \phi_1 - \sin \phi_2)})]^{1/2} \quad (22)$$

We now use the following identity in order to further simplify this equation.

$$e^{j\theta} = \cos \theta + j \sin \theta \quad (23)$$

Also, in an array, the distance between the elements is set equal to one-half of the wavelength, i.e. $d = \lambda/2$. Therefore, equation (22) becomes

$$c = N_1 + N_2 \pm [N_1^2 + 2N_1 N_2 \cos \pi(\sin \phi_1 - \sin \phi_2) + N_2^2]^{1/2} \quad (23)$$

For example: If $\phi_1 = \phi_2$, then (23) becomes

$$c_1 = 2(N_1 + N_2)$$

$$c_2 = 0$$

i.e., the array sees only one noise source

If $\phi_1 = \pi/6$ and $\phi_2 = -\pi/6$, then

$$c_1 = 2N_1 \text{ and } c_2 = 2N_2$$

i.e.,-the array sees two distinct noise sources

The equation shows how the eigenvalues are affected by both the incident angles, ϕ_i , at which the noise signals arrive, the power of these signals, and the spacing, d , of the array elements. It will become apparent that the eigenvalues a_1, a_2 represent independent noise sources as the antenna array output channels see them. The c_1, c_2 do not necessarily have a one-to-one correspondence with the actual sources n_1, n_2 .

V. Examples and Conclusions

We now present several examples which clearly illustrate the concepts that have been presented. the manner in which the parameters will be made to vary is shown in Figure 3.

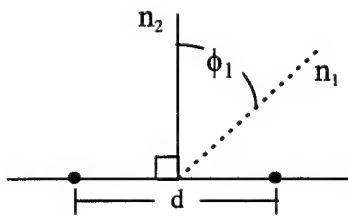


Figure 3. Direction of the Noise Source, n_1 with n_2 fixed

For example: Let us assume that $d = \lambda/2$ and $N_1 = N_2 = 1$. Let us also assume that $\phi_2 = 0$. We will now change the value of ϕ_1 from $-\pi/2$ to $\pi/2$. (see **Figure 4**)

From this graph, we can make two observations. First, we can see that the sum of the eigenvalues is a constant. This constant is equal to twice the sum of the intensities of the noise sources

(N_1 and N_2) arriving at the antenna array. Therefore, we know that the sum of the eigenvalues is independent of the distance between the elements as well as the incident angle of the signals. We are also aware that the eigenvalues are linear and varying combinations of n_1 and n_2 as their difference in direction of arrival, ϕ , changes. they also repeat themselves at regular intervals.

We now observe **Figure 5** in order to determine what will happen to the eigenvalues if ϕ_2 remains constant at $\pi/6$ rather than 0.

From this graph, we observe that the curves have been shifted in phase by $\pi/6$ radians. It is important to note that neither the range nor the wavelength of the graph have changed. This is true for any angle, as shown in **Figure 6** where $\phi_2 = \pi/4$.

Changing the angle of incidence of one of the signals is not the only way to change the eigenvalues. They are also affected by the intensities of the noise sources. In all of the previous examples we have assumed that both intensities were equal to 1. In the following

examples (**Figure 7** and **Figure 8**), we observe that by increasing the intensity of N_1 relative to N_2 , the range is altered and the two eigenvalues cannot be equal.

Now, we shall observe the effect that the distance between the elements has on the eigenvalues. Generally, it is assumed that $d = \lambda/2$ where λ is the wavelength of the interference. However, if this is not the case, then the effect of grating lobes

come into play when $d > \lambda/2$. In **Figure 9**, for example, $d = \lambda$.

In this rather unusual graph the curves for the two eigenvalues overlap and repeat over smaller intervals of direction, ϕ_2 . The repeating occurs when n_1 has been positioned at a grating lobe angle from the location of n_2 in the array pattern. This means that when the distance between the elements is greater than $\lambda/2$, the array cannot distinguish between two signals if they are a grating lobe distance apart. However, since Eigenvalue 1 is defined as

$$N_1 + N_2 + [N_1^2 + 2N_1N_2\cos\pi(\sin\phi_1 - \sin\phi_2) + N_2^2]^{1/2}$$

the graph takes on an unorthodox appearance.

In **Figure 10**, $d = \lambda/3$. This graph yields far different results.

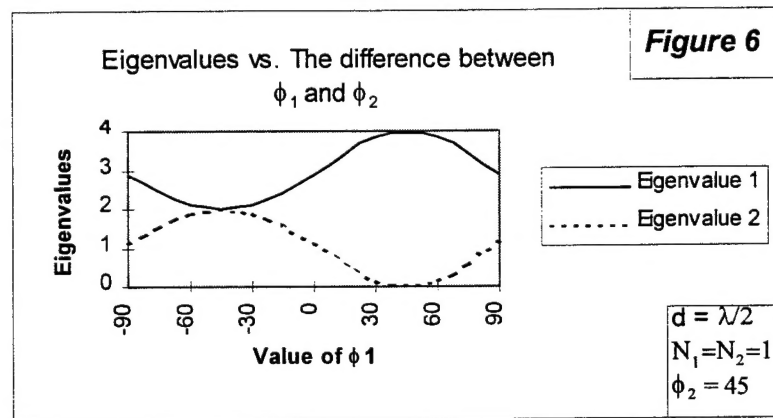
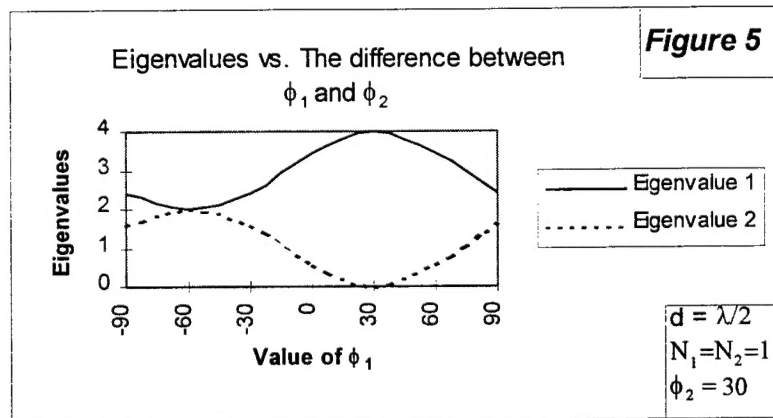
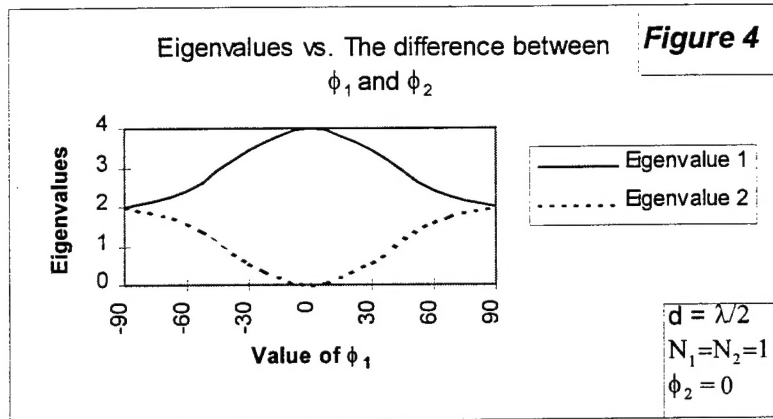
When $d = \lambda/2$, the two eigenvalues don't quite become distinct. The distance between the two elements has decreased beyond the point where the array pattern has distinct lobes. This tends to make the two noise sources appear like one noise source to the array. The array starts to lose its ability to discriminate between the two noise sources. Therefore, the covariance matrix approaches a 1×1 matrix with only one eigenvalue. The following graph(**Figure 10**) is the result when the distance between the elements is zero.

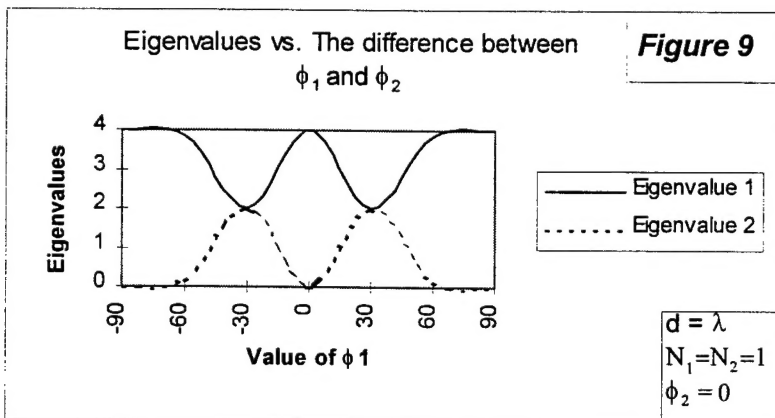
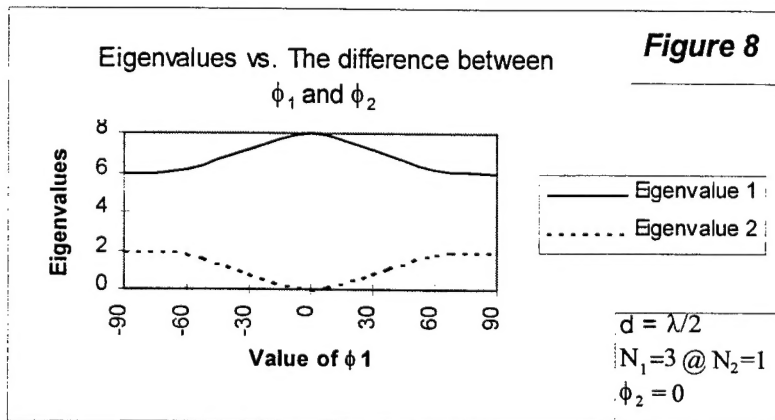
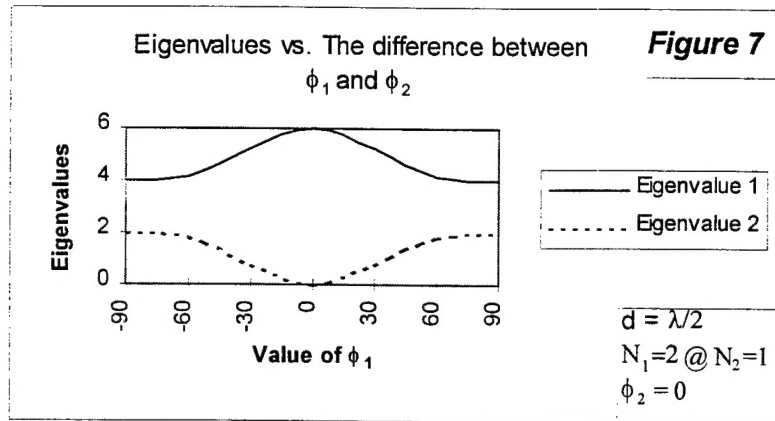
Finally, we assume that $d = \lambda$, that $N_1 = 2$, and that $N_2 = 1$. As shown in **Figure 7**, when the intensities of the two noise sources are not equivalent, the

eigenvalues cannot be equivalent. And as seen in **Figure 9**, when $d > \lambda/2$, the effects of the grating lobes are apparent at both $-\pi/2$ and $\pi/2$ radians; this causes an overlapping of the two curves. Therefore, when we include both of these parameters, the result is **Figure 12**.

As we can see, even though the distance between the two arrays has increased, since $N_1 \neq N_2$, the eigenvalues cannot be equivalent. It is important to note, however, that the grating lobes still affect the eigenvalues.

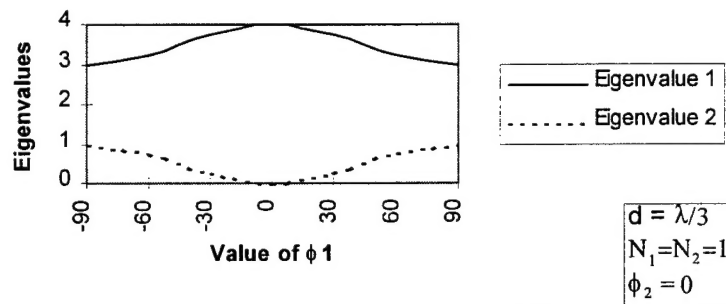
The main point of this paper is this: The physical significance of the eigenvalues play a major role in signal processing. Through the eigenvalues, we are able to determine the relative strengths of the jammers as well as their relative angles of incidence.





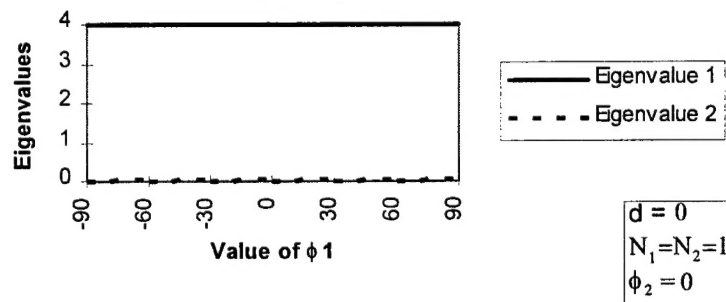
Eigenvalues vs. The difference between
 ϕ_1 and ϕ_2

Figure 10



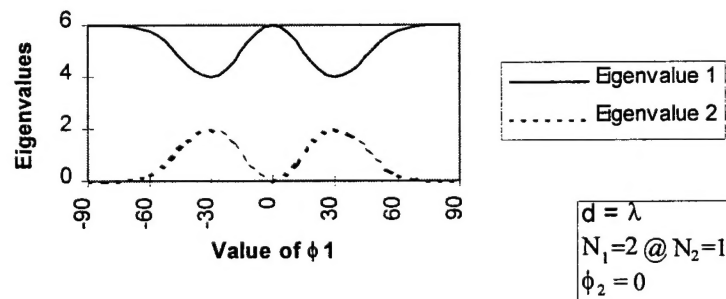
Eigenvalues vs. The difference between
 ϕ_1 and ϕ_2

Figure 11



Eigenvalues vs. The difference between
 ϕ_1 and ϕ_2

Figure 12



References

1. *Adaptive Arrays*, Special Issue, Ieee Trans AP, Vol AP - 24, No 5, September 1976.
2. Reed, L.S. and Brennan, L.E. *Theory of Adaptive Radar*, IEEE Transactions on Aerospace and Electronic Systems, vol. AES-9, No. 2, pp. 237-251, March 1973.
3. Goldstein, J., Williams, D., and Holder, E. *Cross-spectral subspace selection for rank reduction in partially adaptive sensor array processing*, submitted to IEEE signal Processing Letters in June 1994.
4. Wang, H., Park, H.R., and Wicks, M. *On preformance evaluation of space-time processing*, Proc. IEEE Adaptive Antenna Systems Symposium, Melville, NY, Nov. 7-8, 1994.
5. Hudson, J.E. *Adaptive Array Principles*, IEEE Electromagnetic waves series, Vol 11, New York, Institution of Electrical Engineers, 1981.
6. Kolman, Bernard. *Elementary Linear Algebra*, Chapter 6, New York : Macmillan Publishing Co., Inc., second edition, 1977.
7. Van Trees, Harry L. *Detection, Estimation, and Modulation Theory*, Part I, Chapter 2, New York, John Wiley and Sons, 1968.
8. Kraus, John D. *Antennas*, Chapter 4, New York : McGraw-Hill Book Company Inc., 1950.
9. Papoulis, Athanasius. *Probability, Random Variables, and Stochastic Processes* , New York : McGraw-Hill Book Company Inc., 1965.